VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Přehled a implementace vybraných algoritmů používaných pro dělení grafů

Overview and Implementation of Selected Algorithms for Graph Partitioning

2013 Martin Pokluda

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Pokluda**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Přehled a implementace vybraných algoritmů používaných pro dělení grafů.
Overview and Implementation of Selected Algorithms for Graph Partitioning.

Zásady pro vypracování:

Cílem této práce je vytvoření přehledu a implementace několika algoritmů, které se používají pro dělení grafů.

1. Prostudovat a vypracovat přehled algoritmů pro dělení grafů.
2. Vybrat a implementovat několik algoritmů pro dělení rozsáhlých grafů.
3. Provést experimenty s vybranými algoritmy.
4. Porovnat výsledky obdržené různými algoritmy z pohledu časové náročnosti a kvality dělení grafů. Provést srovnání s výsledky obdrženými z programu Metis.
5. Pomocí vybraných nástrojů vizualizovat získané výsledky.

Seznam doporučené odborné literatury:

B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphics", The Bell Sys. Tech.a Journal, vol. 49, no.2, pp. 291-307, 1970.
Elsner, U. Graph partitioning - a survey. MONARCH Dokumenten und Publikationsservice Germany, 2005.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Pavla Dráždilová**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2013 .....

Chtěl bych poděkovat své vedoucí Mgr. Pavle Dráždilové za odborné vedení mé práce, rady, podměty a čas, který mi věnovala.

Abstrakt

Tato diplomová práce se zaobírá tématikou algoritmů pro dělení grafů. V dnešní době tato problematika nachází uplatnění zejména v dopravě, sociálních sítích a v mnoha průmyslových odvětvích. Tato práce si klade za cíl seznámit se s různými druhy algoritmů používaných pro dělení grafů, naimplementovat vybrané druhy algoritmů a otestovat jejich časovou náročnost a kvalitu dělení. Tyto výsledky budou posléze porovnány s výsledky z programu METIS.

Klíčová slova: graf, síť, tok, algoritmus, minimální řez, bisekce, modularita, METIS, Gephi

Abstract

This master thesis is deals with the topic of graph partitioning algorithms. Nowadays, the issue of algorithms is relevant especially in transport, social networks and many industrial sectors. The thesis aims to familiarize the readers with the different types of algorithms used for graph partitioning, implementation of selected types of these algorithms and the test of their time requirements and quality division. These results are then compared with the results from the METIS.

Keywords: graph, mesh, flow, algorithm, minimal cut, bisection, modularity, METIS, Gephi

Seznam použitých zkratek a symbolů

GGGP	–	Greedy Graph Growing Partitioning
F-F	–	Ford-Fulkerson
K-L	–	Kernighan - Lin
GPS	–	Global Positioning System
FEA	–	Finite Element Analysis

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

V dnešní době, kdy se klade velký důraz na kvalitu a rychlost získávání informací, se hledají nejoptimálnější řešení v různých oblastech. Jednou z těchto oblastí, kde se neustále rozvíjejí a testují nové řešení, je obor Teorie grafů. Vyvíjejí se různé optimalizace pro navigační zařízení, paralelizují se výpočty rozsáhlých grafů. Tématem dělení rozsáhlých grafů se zajímá i tato práce.

Tato diplomová práce je zaměřena na řešení problematiky dělení rozsáhlých grafů. Cílem této práce je otestovat vybrané algoritmy na různě velkých grafech (především však rozsáhlých) a z dosažených výsledků vyvodit jejich vhodnost použití na různě velkých grafech.

V druhé kapitole této práce jsou definovány potřebné pojmy spjaté s touto prací. Jsou zde vysvětleny jak grafové, tak i teoretické pojmy z oblasti síti, které jsou v této práci použity při definování a objasňování algoritmů.

V třetí kapitole je popsáno reálné využití grafových algoritmů v praxi. Je zde podrobně popsáno využití v dopravě (GPS navigace), v sociálních sítích, k detekci komunit a k analýze materiálů v průmyslu. Je zde také popsán program Gephi pro vizualizaci grafů, který byl v této práci použit. Také aplikace Metis je zde podrobně popsána. Tato aplikace je použita a výsledky jsou porovnávány s výsledky vlastní implementace algoritmů.

Čtvrtá kapitola se zabývá detailním popisem algoritmů pro dělení grafů. Nachází se zde popis rekurzivní grafové bisekce, algoritmy založené na spektrálním dělení, víceúrovňové algoritmy a algoritmy použity při implementaci testovací aplikace - Ford-Fulkersonův, Kernighan-Linův a Greedy Graph Growing Partitioning algoritmus.

Pátou kapitolu pak tvoří popis metodik, jak měřit kvalitu výsledného rozdělení grafu. Jsou zde popsány různé přístupy a řešení, jak změřit kvalitu výsledného dělení.

Poslední kapitola obsahuje popis testovací aplikace na úrovni tříd. Také jsou zde popsány očekávané výsledky, kterých by se mělo pomocí testů dosáhnout. Dále jsou zde již sepsány naměřené hodnoty prováděných testů a závěrečné vyhodnocení výsledků jednotlivých testů.

2 Úvod do teorie grafů

V této kapitole jsou popsány potřebné informace z oblasti grafů a sítí, jejichž znalost je potřebná k objasnění následujících pojmů a algoritmů v dalších kapitolách.

2.1 Základní pojmy z teorie grafů

V této podkapitole budou popsány základní pojmy z teorie grafů, které budou uplatněny v dalších kapitolách této práce. Informace pro definice vlastností grafů byly čerpány z [10, 16, 17].

- ***Obecný graf G*** je definován jako trojice $G = (V, E, \epsilon)$, kde V je neprázdná množina vrcholů, E je množina hran, $E \cap V = \phi$, a ϵ je incidenční zobrazení:

$$\epsilon : E \to \binom{V}{2} \cup V^2 \cup V$$

- Graf G1 nazveme ***podgrafem*** grafu G, jestliže $V(G1) \subseteq V(G)$ a $E(G1) \subseteq E(G)$.

- ***Orientovaný graf G*** je trojice $G = (V, E, \epsilon)$, kde V je konečná množina vrcholů, E je konečná množina hran a zobrazení ϵ přiřazuje každé hraně z E uspořádanou dvojici vrcholů (u, v).

- ***Neorientovaný graf G*** je trojice $G = (V, E, \epsilon)$, kde V je konečná množina vrcholů, E je konečná množina hran a zobrazení ϵ každé hraně z E přiřazuje jedno nebo dvou prvkovou množinu vrcholů (u, v).

- ***Sledem*** v_0v_n grafu G rozumíme takovou posloupnost vrcholů a hran

$$(v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n),$$

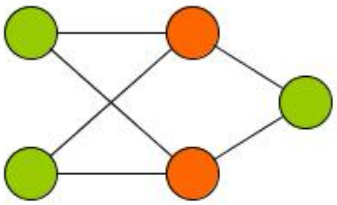
  kde v_i jsou vrcholy grafu G a e_i jsou hrany grafu G, přičemž každá hrana e_i má koncové vrcholy v_{i-1} a v_i. Počet hran nazveme délkou sledu v_0v_n. Vrchol v_0 nazýváme počátečním a vrchol v_n koncovým vrcholem sledu.

- Řekněme, že vrchol v je dosažitelný z vrcholu u, jestliže v grafu existuje sled z vrcholu u do vrcholu v. Graf nazveme ***souvislý***, jestliže pro každé dva vrcholy u, v je vrchol v dosažitelný z vrcholu u. V opačném případě je graf ***nesouvislý***.

- ***Rovinný graf*** je graf, pro který existuje takové rovinné nakreslení, že se žádné dvě hrany nekříží.

- Graf, jehož hrany nebo vrcholy jsou opatřeny číselnými nebo jinými hodnotami, se nazývá ***ohodnocený graf G***, jestliže platí: $w : E \to R$ nebo $V \to R$, kde R je množina reálných čísel.

- ***Nezávislá množina*** v grafu je taková množina jeho vrcholů, že žádné dva z nich nejsou spojeny hranou.



Obrázek 1: Ukázka vrcholů nezávislé množiny; zdroj [19]

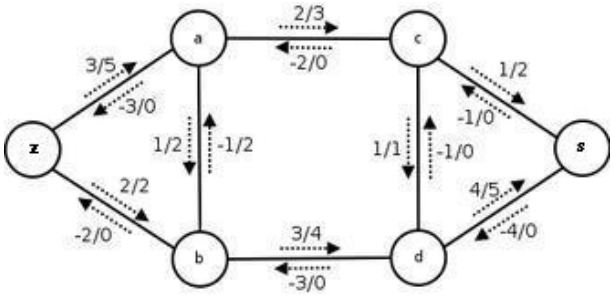
- ***Řez*** je definován jako rozdělení grafu do dvou disjunktních množin. Velikost řezu je součet hran jejichž vrcholu leží v odlišném podgrafu.
- ***Stupeň vrcholu deg(v)*** v grafu G je počet hran, se kterými je vrchol incidentní.
- Graf na n vrcholech, které jsou spojeny po řadě $n - 1$ hranami se nazývá ***cesta*** a značí se P_n. Cestu lze definovat také jako sled, ve kterém se neopakují vrcholy.
- ***Nejkratší cesta*** je taková cesta v ohodnoceném grafu G, kdy z vrcholu u do vrcholu v existuje cesta, jejíž součet ohodnocených hran je nejmenší možný.
- ***Shluk*** můžeme definovat jako množinu objektů, u nichž je hodnota koeficientu podobnosti vyšší, než hodnota jistého prahu.
- ***Komunitou*** můžeme nazvat síť u níž lze vrcholy snadno rozdělit do sady vrcholů, jenž jsou vnitřně hustě propojeny. Obecně platí, že u dvojice uzlů je více pravděpodobné, že budou propojeny mezi sebou, pokud jsou oba členy jedné komunity. Naopak, jestliže oba vrcholy se nenachází ve společné komunitě, je menší pravděpodobnost propojení těchto vrcholů.

2.2 Základní pojmy z teorie sítí

Tato podkapitola se bude věnovat základním pojmům z teorie sítí, toku v nich a dalším důležitým pojmům, které je potřeba znát pro jejich pozdější využití. Tyto informace a definice byly čerpány především z tohoto zdroje [4].

Pojem síť se používá tehdy, když se snažíme pojmenovat matematický model situací, ve kterých se přepravuje hmotná či nehmotná látka po předem daných cestách, které obvykle mají omezenou kapacitu (jedná se např. o silniční dopravu, kdy se přepravuje zboží z místa A do místa B, nebo vodovodní potrubí apod.). Abychom lépe pochopili, co přesně síť znamená, definujme ji takto:

- ***Síť*** je čtveřice S = (G, z, s, w), kde:
  - G je orientovaný graf,
  - vrcholy $z \in V(G), s \in V(G)$ jsou zdroj a spotřebič (někdy také stok),
  - $w : E(G) \to R^+$je kladné ohodnocení hran, zvané kapacita hran.



Obrázek 2: Ukázka sítě; zdroj: [15]

Na obrázku 2 lze vidět příklad jednoduché sítě, která představuje situaci dopravce (zdroj), který převáží své zboží k zákazníkovi (spotřebič). Jednotlivé hrany zde můžou představovat úseky silnic, železnic, silniční uzly, železniční uzly, překladiště apod. Ohodnocení hran pak může znamenat cenu jednotlivých úseků přepravy.

- Mějme ohodnocený graf G. ***Tokem v síti G*** nazýváme takové ohodnocení hran reálnými čísly $f : E(G) \to R$, které pro každý vrchol v, splňují Kirchhoffův zákon:

$$\sum_{e \in E^+(v)} f(e) = \sum_{e \in E^-(v)} f(e)$$

Graf (viz obrázek 2) se dá představit jako soustava potrubí, kterým proudí nějaká kapalina. Tok v každé z hran je představován jako ustálený a beze ztrát. Kirchhoffův zákon pak říká: „Kolik tekutiny do vrcholu přiteče, tolik z něj taky odteče“. Orientace hran v grafu říká kudy daný tok (kapalina) poteče. Hrana může být ohodnocená taky záporně, v tom případě to znamená, že tok poteče obráceně, tedy v opačném směru hrany.

- Velikost toku f (e) v jednotlivých hranách bývá většinou omezena dolním a vrchním minimem resp. maximem. Platí tedy: $f(e) \in \langle l(e), c(e) \rangle$, kde číslo $l(e)$ se nazývá dolním omezením toku a číslo $c(e)$ se nazývá horním omezením toku. Tok $f(e)$, který splňuje nerovnosti $l(e) \leq f(e) \leq c(e)$, se pak nazývá ***přípustným tokem***.

- ***Velikost toku od zdroje ke spotřebiči*** se definuje jako množství toku, které vzniká ve zdroji, tj. jako rozdíl:

$$F(f) = \sum_{e \in E^+(z)} f(e) - \sum_{e \in E^-(z)} f(e)$$

Nyní nadefinujeme velikost toku přes řez, kdy tato znalost bude potřeba např. ve Ford- Fulkersonově algoritmu (viz kapitola 4.6).

- ***Velikost toku přes řez*** určený množinou A definujeme jako množství toku, které hranami řezu teče z množiny A ven, zmenšené však o množství, které se hranami řezu zpět do množiny A vrací:

$$F_A(f) = \sum_{e \in W^+(A)} f(e) - \sum_{e \in W^-(A)} f(e)$$

Věta: Nechť f je libovolný tok od zdroje z ke spotřebiči s a nechť $W_{(A)}$ je libovolný řez, který odděluje zdroj a spotřebič. Potom platí $F_A(f) = F(f)$, tj. velikost $F(f)$ toku od zdroje ke spotřebiči je rovna velikosti $F_A(f)$ toku přes řez $W_{(A)}$.

Teď, když víme, že velikost toku přes řez je rovna velikosti toku od zdroje ke spotřebiči, můžeme nadefinovat kapacitu řezu takhle:

- ***Kapacitu řezu*** $W_{(A)}$ definujeme jako číslo:

$$C_A = \sum_{e \in W^+(A)} c(e) - \sum_{e \in W^-(A)} l(e)$$

Tato definice jasně říká, že tok od zdroje ke spotřebiči nemůže být větší, než je velikost toku přes řez (za podmínky, že řez odděluje spotřebič od zdroje).

Dalšími důležitými pojmy, kterými se budeme zaobírat, jsou maximální tok v síti a minimální řez síti.

- ***Minimální řez*** v síti je takový řez síti, kdy součet kapacit množiny hran řezu je minimální možný, aby bylo zajištěno rozdělení sítě do dvou disjunktních množin.
- ***Maximálním tokem*** v síti je myšleno nalezení největšího možného toku v síti od zdroje z k spotřebiči s vzhledem k ohodnocení hran dané sítě.

Abychom mohli popsat algoritmus, který jednoznačně „řekne“, jak postupovat při hledání maximálního toku v síti, musíme znát definici a postup při hledání tzv. *zlepšující cesty*.

- ***Zlepšující cesta*** vzhledem k toku f je taková neorientovaná cesta ze zdroje z do spotřebiče s, jejíž každá hrana e splňuje:

Je-li e hranou vpřed, pak $f(e) < c(e)$,
Je-li e hranou vzad, pak $f(e) > l(e)$.

Hrana se nazývá *hranou vpřed*, je-li orientována ve směru průchodu cestou. *Hranou vzad* je pak myšlena hrana, která je orientována v protisměru průchodu cestou. K zlepšující cestě se úzce váže pojem *kapacita zlepšující cesty*. Tímto pojmem je myšlena maximální hodnota d, o kterou lze změnit tok na zlepšující cestě. Maximální hodnota je tedy:

$$d = min(c(e) - f(e), f(e) - l(e)).$$

Vybírá se nejmenší hodnota z těchto dvou rozdílů, ta je pak brána jako maximální hodnota d. Známe-li již zlepšující cestu a maximální hodnotu, už nám nic nebrání v úpravě sítě.

Postup je následující:

- u hran vpřed se tok zvýší o hodnotu d.
- u hran vzad se tok sníží o hodnotu d.

Tímto jednoduchým postupem se upraví síť (úpravy sítě pouze na místech zlepšující cesty) a zvýší se tak tok o hodnotu d. Tímto však úprava sítě nemusí končit. Můžeme se pokusit nalézt další zlepšující cestu v síti a zvýšit tak již jednou upravenou síť o další maximální hodnotu zlepšující cesty.

Nyní můžeme popsat postup, jak tyto zlepšující cesty v dané síti nalézt. K hledání zlepšující cesty se používá tzv. značkovací procedura. Postup je následující:

Algoritmus 1: Značkovací procedura
Vstup: *graf* $G = (V, E)$
Výstup: *zlepšující cesta*

1. Označí se vrchol (zdroj z), ostatní zůstávají beze značek.
2. Existuje-li hrana e taková, že platí $P_{v(e)}$ (hrana počátečního vrcholu) má značku, $K_{v(e)}$ (hrana koncového vrcholu) nemá značku a platí $f(e) < c(e)$, pak se označkuje $K_{v(e)}$.
3. Existuje-li hrana e taková, že platí $K_{v(e)}$ má značku, $P_{v(e)}$ nemá značku a platí $f(e) > c(e)$, pak se označkuje $P_{v(e)}$.
4. V momentě, kdy je označkován spotřebič s, je hotovo, značkování končí a zlepšující cesta je nalezena (označené vrcholy).

   Není-li však koncový bod (spotřebič s) označkován a nelze-li již označkovat žádný jiný vrchol, zlepšující cesta již v dané síti neexistuje.

Pomocí tohoto postupu se jednoznačně nalezne/nenalezne zlepšující cesta v síti. Naleznou-li se všechny zlepšující cesty v dané síti, nalezne se i maximální tok a minimální řez sítě.

3 Oblasti použití jednotlivých typů sítí

Oblast využití sítí, jakožto i celé Teorie grafů v praxi, je rozsáhlá. Od základních matematických úloh, které se učily již na základní škole, až po využití v dopravě, telekomunikaci, informatice, ekonomice, energetice, chemickém průmyslu apod. Nejlépe si lze představit využití Teorie grafů v dopravě, například, chceme-li převézt určité množství zboží z místa A do místa B. Pomocí Teorie grafů se může, za použití jistých metod a algoritmů, určit jaká trasa pro přepravu bude nejlevnější, nejrychlejší, nejméně nákladná, kolik zboží lze převézt z místa A do místa B po určité trase. Jsme také schopni navrhnout, kde je pro firmu nejvýhodnější např. postavit skladiště, depo, odpočinkové místo pro řidiče apod. Získá se tak nespočet důležitých informací.

Z hlediska využití dělení grafů v praxi se naskýtá mnoho oblasti použití. V dopravě se dělení grafů využívá především pro zkracování (viz 4.3) dopravní sítě. Vytváří se zjednodušený model obcí a měst, nad kterými se provádí výpočty (nejkratší cesta, nejrychlejší cesta apod.), které jsou následně promítnuty do původní mapy. Pomocí tohoto rozdělení grafu lze značně urychlit výpočet - lze paralelizovat výpočet (např. nejkratší cesty v městech se mohou počítat současně vzhledem k celkové trase - procházející např. několika městy).

V sociálních sítích se dělení grafů využívá nejčastěji k detekci tzv. komunit (viz 3.3.1). Měří se provázanost mezi jednotlivými komunitami, vrchol s nejvyšším stupněm - centrum komunity, nejdelší cesta v komunitě apod. Tyto naměřené hodnoty jsou následně dále podrobněji rozebírány a analyzovány.

V neposlední řadě se dělení grafů používá v průmyslových odvětvích a to především pro paralelizaci výpočtů vlastnosti materiálů. Materiálové vlastnosti jsou popsány a namodelovány pomocí speciálního druhu sítě - mesh (viz 3.2). Cílem je rozdělení této sítě do podoblastí, kdy každá tato podoblast může být namapována na jiný procesor. Tyto oblasti musí být rozděleny tak, aby komunikace mezi jednotlivými sub oblastmi byla minimalizována.

Tyto oblasti z Teorie grafů, která našla praktická uplatnění v každodenním životě, budou nyní podrobněji představeny.

3.1 Dopravní sítě

Jak již bylo řečeno, důležitou roli hraje Teorie grafů v dopravě. Asi nejznámějším dopravním zařízením, používané v dnešní době, je ***GPS (Global Positioning System) navigace.*** Následující informace byly čerpány z [1].

Hlavním problémem GPS navigací je ten, jak zjednodušit mapy cest vzhledem k paměti přístroje. Zejména při vyhledávání nejkratší cesty může být náročnost výpočtu větší, než je kapacita paměti přístroje, vzhledem k množství cest vedoucích z místa A do požadovaného místa B. K tomuto účelu se využívá minimální řez a maximální tok v síti.

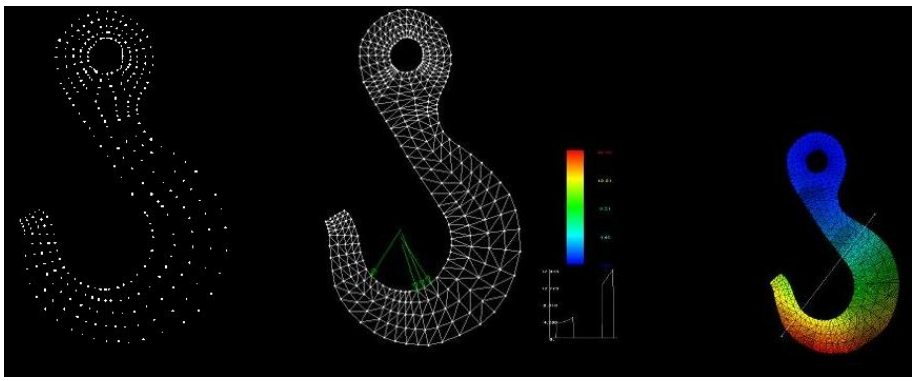
Z původní mapy (grafu) obsahující tisíce vrcholů a hran se vytvoří multigraf, kdy jednotlivé vrcholy a hrany jsou nahrazovány podle daných pravidel. Některé objekty v mapách chceme, aby byly nedělitelné, některé naopak nahrazujeme. Například silniční síť města či obce můžeme nahradit jedním vrcholem a hrany, které původně vycházely

z tohoto města do obcí a měst zachováme, případně, vede-li více hran z jednoho města do druhého, můžeme tyto hrany nahradit jednou hranou o dané kapacitě (například nejkratší vzdálenost). Dále je snaha o eliminování smyček v grafu, které vznikají odebíráním vrcholů při zjednodušování. Je-li mapa, takto zjednodušená, stále nevyhovující, lze ji dále zjednodušovat, přičemž musí být zajištěno, že se lze z těchto stupňů zjednodušení vrátit zpět do původní mapy, jelikož se požaduje nalezení nejkratší cesty v reálné mapě a ne v zjednodušené verzi. Všechny stupně zjednodušení si nesou sebou ohodnocení odpovídající úpravám a reálným hodnotám z původní mapy. Díky tomu lze určit, jen za pomocí hran a vrcholů, nejkratší cesta. Uživateli jsou pak nabídnuta tato data a především zobrazení mapy v různých stupních zjednodušení.

3.2 Síť (mesh) pro metodu konečných prvků (Finite element method)

Pod pojmem *mesh* se dá představit rovinný graf, jehož vrcholy jsou „hustě“ propojeny s ostatními vrcholy dle daných charakteristik (můžou objekt dělit na trojúhelníkové nebo čtvercové podsítě), tak jak ukazuje obrázek 3, kde objekt je rozdělen na trojúhelníkové podsítě. Pojem mesh je především spjat s pojmem analýza konečných prvků (FEA). Jde o modelování výrobků a systémů ve virtuálním prostředí, pro účely zjištění a řešení strukturálních či výkonnostních problémů. FEA je praktická aplikace metody konečných prvků, kdy pomocí matematického modelu lze detekovat velmi složité strukturální problémy. Analýzu konečných prvků lze použit v mnoha průmyslových odvětvích, avšak nejčastěji se používá v leteckém, biomechanickém a automobilovém průmyslu.

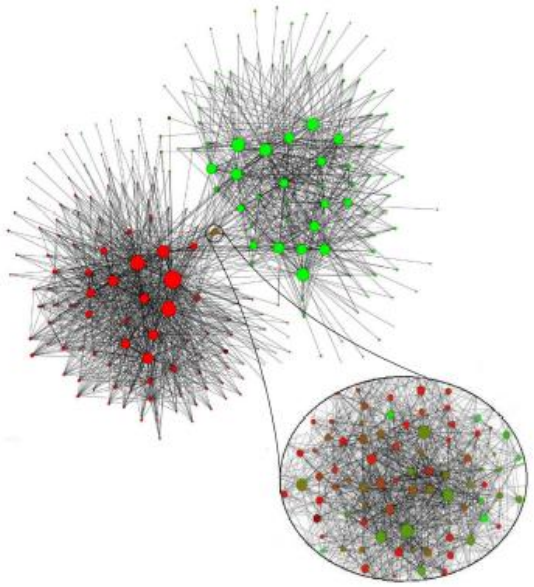
Model konečných prvků zahrnuje systém bodů (uzlů), které tvoří tvar konstrukce. K těmto uzlům jsou konečné prvky připojeny a obsahují materiální a strukturální vlastnosti daného modelu. Pomocí těchto vlastností lze následně určit, jak bude materiál reagovat například na změnu namáhání v určité oblasti. Oblasti, které jsou nejvíce pod tlakem (namáháním) vyžadují obvykle hustší síť, naopak méně namáhané části jsou obvykle tvořeny řidší síti. Oblasti s hustou sítí jsou většinou oblasti zájmu, kdy je snaha určit „nejproblémovější “ místo výrobku. Většinou v nějakém zaoblení, v rozích, po zátěžových zkouškách v namáhaných oblastech. [12]



Obrázek 3: Ukázka mesh sítě; zdroj: [11]

3.3 Sociální sítě

Již od svého narození si člověk začíná vytvářet svou vlastní síť. Nejdříve si vytváří vazbu na své rodiče a členy rodiny, postupem času se k nim přidávají vazby na spolužáky, učitele ze školy, přátelé, kolegy z práce. Takto vytvořenou síť můžeme nazvat sociální síti. Sociální síť je reprezentována pomocí grafu. Vrcholy představují jednotlivé osoby, přátelé, kolegy apod. Hrany mezi nimi představují vztah, jaký k sobě dané osoby mají (spolužák, otec, matka, kolega z práce, ale také komunikaci mezi sebou, pracovní pozice v dané firmě apod.). K určení intenzity (váhy) těchto vazeb mezi jednotlivými vrcholy se používá síla, váha hrany. Například k nejlepšímu kamarádovi bude člověk mít váhu vazby 5 (použijeme-li například stupnici od 1-5, kdy 1 je nejmenší váha) a třeba spolužák ze školy jen 2.



Obrázek 4: Ukázka komunitní struktury u mobilního operátora v Belgii; zdroj [3]

Na obrázku 4 lze vidět shlukové struktury dle používání řeči při telefonním hovoru u jednoho z mobilních operátorů v Belgii k roku 2008. Na obrázku jsou vidět dva shluky. Červený, ten tvoří lidé mluvící Francouzsky a zelený jsou lidé mluvící Vlámsky. Tyto dva shluky představují 85% všech volajících v Belgii. Uprostřed těchto dvou shluků se nachází jeden malý shluk (přiblížený). V tomto shluku se nachází lidé, kteří hovoří oběma jazyky.

Sociální sítě se většinou definují jako prostor, kde lidé spolu komunikují, tvoří mezi sebou vazby a sdílejí informace mezi sebou. V dnešní době jednou z nejznámějších online sociálních sítí na světě je Facebook. Tuto síť tvoří lidé (vrcholy) a přátelství (vazby) mezi nimi. V této síti existují různé shluky lidí (skupiny), kteří spolu sdílejí stejnou myšlenku, záměr, vyznání. Takovýmto shlukům v sociálních sítích se říká **„komunity“**. Lidé zde sdílejí své pocity, zájmy, fotky, videa a mnohé další informace. Hlavním cílem

této sítě je umožnit komunikaci mezi sebou, zábavu, ale také je to jeden z nejsilnějších marketingových nástrojů, sloužících k reklamě, získávaní tzv. feedbacků a statistik pro firmy a produkty. Facebook k únoru 2012 čítá okolo 900 mil. aktivních uživatelů.

Tedy díky znalosti sociálních sítí se může lépe vystihnout šíření informací, shlukování lidí, jevy v ekonomii a mnoho dalších informací.[3]

3.3.1 Detekce komunit v sociálních sítích - obecný pohled

Informace pro tuto podkapitolu byly čerpány z tohoto zdroje [3]. Komunitou lze nazvat například skupina lidí mající oblíbený jeden fotbalový klub vzhledem ke všem ostatním fotbalovým klubům. Představíme-li si každého člověka z této skupiny jako vrchol v grafu a hrany budou reprezentovat vztah známosti mezi sebou, pak určitě bude existovat více takovýchto hran v dané komunitě (v jednom fotbalovém klubu), než hran směřujících mezi komunitami. Stručně řečeno existuje víc vazeb v jedné komunitě než mezi jednotlivými komunitami. Tento předpoklad patří mezi základní podmínky definice komunit.

Nyní bude tato skutečnost definována po stránce matematické. Pomocí tohoto postupu (výpočtu) jednoznačně určíme, zda vybraný podgraf grafu tvoří komunitu či nikoliv:

Mějme podgraf C grafu G, tedy $\mid C \mid = n_c, \mid G \mid = n$. Definujme **vnitřní hustotu shluku** $\sigma_{int}(C)$ podgrafu C jako poměr mezi počtem vnitřních hran podgrafu C a počtem všech možných vnitřních hran podgrafu C, tedy $\sigma_{int}(C) = \frac{I_c}{n_c(n_c-1)/2}$, kde I_c je počet vnitřních hran. Nápodobně definujme **vnější hustotu shluku** $\sigma_{ext}(C)$ podgrafu C jako poměr mezi počtem hran směřujících z vrcholů podgrafu C do vrcholů zbytku grafu G a počtem všech možných hran směřujících z podgrafu C do zbytku grafu G, tedy $\sigma_{ext}(C) = \frac{E_c}{n_c(n-n_c)}$, kde E_c je počet hran směřujících z podgrafů C ven.

Podgraf C se nazve komunitou tehdy, jeli $\sigma_{int}(C)$ znatelně větší, než průměrná hustota propojení σ_G grafu G, která je dána poměrem mezi počtem hran a maximálním možným počtem hran grafu G. Také ale musí platit, že $\sigma_{ext}(C)$ musí být mnohem menší než σ_G. Hledáním nejlepšího kompromisu mezi maximálním $\sigma_{int}(C)$ a minimálním $\sigma_{ext}(C)$ je hlavním cílem všech shlukovacích algoritmů.

Existují další podmínky, které musí podgraf splňovat, aby se stal komunitou. Vždy závisí na nás, jaká pravidla pro detekci komunit stanovíme. Mohou to být tyto:

1. Komunitou může být pouze tzv. „klika“. Klikou označujeme úplný podgraf (v kterém jsou všechny vrcholy spolu vzájemně propojeny). Hledání klik v grafu patří mezi NP-úplné problémy.

2. Každý vrchol v podgrafu musí být v sousednosti s minimálně k vrcholy.

3. Maximální podgraf, ve kterém každý vrchol musí sousedit s ostatními, vyjímá k vrcholů.

4. Omezení na k hran/vrcholů v komunitě.

Nejlépe je hledat takový podgraf, který má minimální řez, tedy podgraf z kterého vystupuje co nejméně hran z vrcholů do zbytku grafu.

Výpočetní složitost
Výpočetní složitost dělení grafů je dána počtem vrcholů n a počtem hran m. Výpočetní složitost algoritmu nemůže být vždy určena. Zápis $O(m^{\alpha}n^{\beta})$ znamená, že výpočetní doba roste exponenciálně s počtem vrcholů a hran.

Algoritmy s polynomiální složitostí patří do třídy P. Pro některé důležité řešení a optimalizační úlohy nejsou dosud známy polynomiální algoritmy. Nalezení složitosti problému v jeho nejhorším možném případě může zabrat čas, který roste rychleji než jakékoliv polynomické funkce. Problémy, jejichž řešení může být ověřeno v polynomiálním čase, spadají do třídy NP, která obsahuje problémy třídy P. Problém je NP-těžký, jestliže může být přepsán do některého z NP problémů. Nicméně NP-těžký problém nemusí spadat do třídy NP. Skutečnost, že NP problém má řešení, které je ověřitelné v polynomiálním čase neznamená, že NP problém má polynomiální složitost, že jsou v P. NP-těžké problémy nemusí být v NP, ale jsou alespoň tak náročné jako NP úplné problémy, takže je nepravděpodobné, že mají polynomiální složitost, přesto důkaz, který by to potvrzoval, stále chybí.

Mnoho algoritmů pro dělení grafů anebo problémů týkajících se dělení grafů jsou NP-těžké. V tomto případě je zbytečné používat „přesné“ algoritmy, které by bylo možno použit pouze v malých grafech. Kromě toho, když má algoritmus polynomiální řešení, může být toto řešení na velkém grafu stále příliš pomalé. Proto se raději na rozsáhlé grafy, sítě používají aproximační algoritmy. Jsou to řešení, které na jednu stranu neposkytnou přesný výsledek (pouze potřebný), na druhou stranu však poskytnou výsledek s přijatelnou časovou složitostí. Aproximační algoritmy se běžně používají pro optimalizační problémy, u kterých se požaduje nalezení minimální nebo maximální hodnoty.

3.4 Dělení grafů

Problém při dělení grafu je ten, jak rozdělit daný graf do skupin o definované velikosti tak, aby byl součet hran směřujících z jedné skupiny do druhé co nejmenší. Takovýto součet nazýváme minimální řez. Některé algoritmy vyžadují, aby byl definován počet skupin. V opačném případě by se mohlo totiž stát, že by algoritmus vyhodnotil celý graf jako jednu skupinu a řešení by pak bylo triviální. Důležité je také určit velikost takové skupiny, jelikož by se mohlo stát, že by algoritmus oddělil vrchol o nejnižším stupni od zbytku grafu, což by nemělo prakticky žádný význam.

Ve většině případů dělení grafů se graf dělí na dva podgrafy pomocí daného algoritmu a iterací do určité hloubky se dále „půlí“ na další menší podgrafy. Takovýto postup bude využit i v této práci za pomoci Ford-Fulkersnova algoritmu (viz kapitola 4.6).

Algoritmy pro dělení grafů nejsou vhodné pro detekci komunit, protože se většinou vyžaduje počet a také velikost dané komunity. Tyto informace jsou většinou neznámy. Kromě toho, iterační dělení grafů do více částí není ideální postup. Jeli, například požadováno rozdělení grafů do tří komunit, většinou se stává, že se původní graf rozdělí do

dvou komunit, přičemž třetí je dosaženo za pomocí minimálního řezu, kdy třetí komunita se skládá z částí z původních dvou komunit.

3.5 Nástroje pro tvorbu, analýzu, dělení a vizualizaci grafů

V této podkapitole budou popsány nástroje METIS a GEPHI, které byly použity v této práci k vizualizaci grafů a testování [13, 14].

3.5.1 Gephi

Jedná se o nástroj, který je distribuován jako open source aplikace. Slouží k interaktivní vizualizaci a průzkumu všech druhů sítí, komplexních systémů, dynamických a hierarchických grafů.

Gephi nabízí tyto možnosti:

Průzkumná analýza dat - intuitivně orientovaná analýza sítí pracující v reálném čase.
Směrová analýza - odhaluje základní struktury sdružení mezi objekty, zejména v rozsahu volných sítí.
Analýza sociální sítě - snadné vytváření sociálních datových konektorů do komunitních map, organizací a malých světových sítí.
Biologické síťová analýza - zastupování vzorů biologických dat.

Gephi také nabízí spoustu možností rozložení výsledného grafu - rozložení vrcholů dle měřítka, vzdálenosti, velikosti prostoru v kterém se mají zobrazit apod.

Další výhodou tohoto nástroje je možnost měřit kvalitu navrženého grafu. Lze testovat celou síť anebo jen vrcholy či hrany - lze měřit modularita sítě, průměrná délka cesty, průměrný stupeň sítě a další.

V této práci byl tento nástroj (konkrétně verze 0.8.2 beta) použit k vytvoření vstupních dat a především k vizualizaci výsledků náhodných, neorientovaných a ohodnocených grafů. K uložení dat byl použit soubor typu GDF. Tento typ souboru byl vybrán především z důvodu jeho nenáročné struktury uložení dat, jelikož bylo zapotřebí převést tento soubor do testovací aplikace a výsledek opět zapsat do GDF souboru.

Více o tomto programu viz [13].

3.5.2 METIS (Family of Graph and Hypergraph Partitioning Software)

METIS je sada programů pro sériové dělení grafů, rozdělení konečných prvků. Algoritmy, realizované v této aplikaci, jsou založeny na víceúrovňové rekurzivní bisekci, víceúrovňovém K-way dělení grafu, a multi-constraint dělících systémech. METIS lze naimportovat do Visual studia po předešlé kompilaci v programu CMake (kompilátor jazyka C, v kterém je METIS napsán). Autorem této aplikace je katedra informatiky a výpočetní techniky v čele s G. Karypisem z univerzity v Minesotě. Tato aplikace se od svého vzniku v roce 1997 neustále vyvíjí.

METIS lze použit k dělení souvislých, řízených i neřízených, ohodnocených i neohodnocených grafů. Tuto aplikaci lze také použit pro dělení mesh. METIS se skládá z několika tříd, nejzajímavější jsou tyto dvě:

gpmetis - slouží k rozdělení grafu (pouze grafu) na požadovaný počet části pomocí víceúrovňového K-way dělení grafu a nebo pomocí víceúrovňové rekurzivní bisekce. Jelikož se jedná o víceúrovňové dělení, lze si zde vybrat další algoritmy, které budou součástí víceúrovňového dělení grafu.

mpmetis - slouží k rozdělení mesh (pouze mesh) na požadovaný počet části.

V této práci byla tato aplikace (konkrétně verze 5.1.0) použita k dělení daných grafů pomocí předem vybraných algoritmů (pomocí třídy gpmetis) a výsledné grafy porovnány s výsledky vlastní implementace. METIS používá vlastní formát souboru pro načítání (soubor.GRAPH) a vlastní formát výstupu (soubor.4 - číslo podle počtu předem zvolených částí pro dělení grafu). Pro ověření kvality a vizualizaci těchto výsledků bylo tedy zapotřebí převést tyto formáty do testovací aplikace a z ní následně do GDF souboru.

Více informaci o této sadě algoritmů a popisu tříd viz [14].

4 Přehled algoritmů pro dělení grafů

Tato kapitola se věnuje přehledu algoritmů, které se nejčastěji používají pro grafové dělení. Je zde prezentován základní nejjednodušší algoritmus pro dělení grafu - rekurzivní grafová bisekce. Dále jsou zde popsány metody spektrálního dělení grafu, víceúrovňové grafové algoritmy a to jak pro spektrální dělení tak i příklad kombinace více algoritmů (Maximální párování + Ford-Fulkersonův algoritmus). Posledními prezentovanými algoritmy jsou pak implementované algoritmy v testovací aplikaci - Ford-Fulkersonův, Kernighan-Linův a GGGP algoritmus. Pro tuto podkapitolu byly informace čerpány z [22]

Dělení grafů
Mějme graf $G = (V, E)$, který je neorientovaný, spojitý a má n vrcholů. Dělení grafu představuje rozdělení grafu G do dvou disjunktních množin A a B se specifickými vlastnostmi. Například K-way dělení grafu rozdělí původní graf G do k přibližně stejných částí. Dále může být definován maximální nebo minimální počet vrcholů, které jednotlivé podgrafy smí obsahovat. Hlavním hodnoticím kritériem dobrého rozdělení grafu je počet a váha jednotlivých hran mezi podgrafy (čím míň, tím lépe).

4.1 Rekurzivní grafová bisekce

Na vstupu tohoto algoritmu je neorientovaný, souvislý graf G. Tento algoritmus nejprve najde dva vrcholy od sebe nejvzdálenější (tato vzdálenost je označována jako průměr grafu). Pak polovina vrcholů ležících nejblíže k jednomu z těchto vrcholů tvoří podgraf A a zbytek tvoří podgraf B. Tento proces se pak opakuje na každém podgrafu.

Pro nalezení dvou vrcholů, které jsou nejdále od sebe, může být použit heuristický postup. Nejprve se označí vrchol R. Jeho sousedé se označí numerickou hodnotou 1, sousedé těchto sousedů pak 2 atd. Nejvzdálenější prvek od vrcholu R má pak hodnotu m. Tento vrchol se označí jako R a postup se opakuje. Pokud se m nemění, jsou tyto vrcholy od sebe nejvzdálenější.

Rekurzivní grafová bisekce má časovou složitost $O(n)$.

4.2 Spektrální dělení grafu

Všechny algoritmy, které v této sekci budou uvedeny, využívají druhý vlastní vektor k půlení grafu. Tedy pomocí tohoto vektoru rozdělí graf G na dva stejně velké podgrafy $G1$ a $G2$, kde $G = G1 \cup G2$. Další dělení podgrafů lze pak volat rekurzivně. V podstatě algoritmy hledají nejmenší možný seznam hran $E' \subseteq E$, které je potřebné odstranit k rozdělení gafu G, případně daného podgrafu.

4.2.1 Spektrální bisekce

Mějme graf $G = (V, E)$ a vektor $\vec{f} = \{v_0, v_1, \ldots, v_n\}$ je druhý vlastní vektor - tzv. Fiedlerův vektor Laplaceovy matice L grafu G. Jestliže je G spojitý graf, pak násobnost nulového

vlastního čísla je 1 a násobnost druhého vlastního čísla je větší než 0. Myšlenkou spektrální bisekce je nalézt medián M vektoru $\vec{f}$, pak můžeme oddělit vrcholy do dvou podgrafů G1 a G2 s ohledem na vyhodnocení Fiedlerova vektoru. Pro vrcholy v podgrafu G_1 platí, že $f_i \leq M$ a podgraf G_2 tvoří zbývající vrcholy. Toto rozdělení se nazývá Fiedlerův řez. Existuje-li jen jeden vrchol rovnající se mediánu, pak patří do podgrafu G_1. Existuje-li více vrcholů rovnající se mediánu, rozdělení do podgrafů je libovolné.

Mějme dvě množiny vrcholů V_1, V_2. Definujme $V_1^{'}$ obsahující vrcholy z V_1, které mají společnou hranu s vrcholem z V_2. Mějme $V_2^{'}$ obsahující vrcholy V_2, které mají společnou hranu s vrcholem V_1. Mějme množinu hran E_1 označující hrany, které mají oba koncové vrcholy v V_1. Mějme množinu hran E_2 označující hrany, které mají oba koncové vrcholy v V_2. Nechť $E^{'} \subseteq E$ je množina hran, které mají jeden vrchol ve $V_1^{'}$ a druhý v $V_2^{'}$. Společně $V_1^{'}$ a $V_2^{'}$ jsou množiny vrcholů obsahujících hrany $E^{'}$ potřebné k rozdělení grafu G do dvou podgrafů G_1 a G_2.

Algoritmus 2: Spektrální bisekce
Vstup: *graf* $G = (V, E)$
Výstup: *grafy* $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$

Krok 1: Výpočet Fiedlerova vektoru $\vec{f}$,
Krok 2: nalezení mediánu M,
Krok 3: foreach (vrchol v grafu G)
if $\vec{f} \leq$ M
přidej vrchol v do V_1
else
přidej vrchol v do V_2
Krok 4: if $\|V_1\| - \|V_2\| > 1$ přesuň některé vrcholy rovné mediánu, aby rozdíl byl 1 a menší.
Krok 5: Nechť $V_1^{'}$ je množina vrcholů z V_1, které sousedí s vrcholy V_2.
Nechť $V_2^{'}$ je množina vrcholů z V_2, které sousedí s vrcholy V_1.
Nechť $E^{'}$ je množina hran, kde jeden z vrcholů leží v $V_1^{'}$ a druhý v $V_2^{'}$.
Krok 6: Nechť E_1 je množina hran, která má oba vrcholy v V_1.
Nechť E_2 je množina hran, která má oba vrcholy v V_2.
Sestroj grafy $G1 = (V1, E1), G2 = (V2, E2)$.
Krok 7: Konec.

4.2.2 K-way partitioning

Mějme graf $G = (V, E)$, kde počet vrcholů je n. Hlavní princip tohoto dělení spočívá ve využití rekurzivní bisekce, kdy graf G se postupně dělí na podgrafy do požadovaného množství - k-podgrafů. Tedy k vyjadřuje počet výsledných podgrafů grafu G.

Při tomto dělení můžou nastat dvě situace. Buď je k sudé a tedy požadovaného počtu se dosáhne iteračním půlením anebo je požadovaný počet lichý a využije se tzv. *kvantil* Q, který představuje požadovanou velikost (vyjádřenou v procentech) jednotlivých pod-

grafů.

Algoritmus 3: Modifikovaná spektrální bisekce
Vstup: *graf* $G = (V, E)$*, kvantil* $Q(\%)$
Výstup: *grafy* $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$

Krok 1: Výpočet vlastního vektoru $\vec{f}$
Krok 2: vyhledej Q ve vektoru $\vec{f}$
Krok 3: foreach (vrchol v grafu G)
if $\vec{f}(n) \leq$ Q
přidej vrchol v do V_1
else
přidej vrchol v do V_2
Krok 4: if $\|V1\| - \|V2\| > 1$ přesuň některé vrcholy rovné mediánu, aby rozdíl byl 1 a menší.
Krok 5: Nechť $V_1^{'}$ je množina vrcholů z V1, které sousedí s vrcholy V2.
Nechť $V_2^{'}$ je množina vrcholů z V2, které sousedí s vrcholy V1.
Nechť $E^{'}$ je množina hran, kde jeden z vrcholů leží v $V_1^{'}$ a druhý v $V_2^{'}$.
Krok 6: Nechť E_1 je množina hran, která má oba vrcholy v V_1.
Nechť E_2 je množina hran, která má oba vrcholy v V_2.
Sestroj grafy $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$.
Krok 7: Konec.

4.3 Víceúrovňové grafové dělení (Multilevel Graph Partitioning)

Hlavní myšlenkou víceúrovňového dělení je získat hrubé přiblížení k požadovanému vlastnímu vektoru a pak použít jiné vhodné iterační schéma k získání konečného vlastního vektoru za použití hrubé aproximace z předchozího kroku. [23]

Jelikož Lanczosova metoda pro získání vlastního vektoru je velmi časově náročná (vzhledem k počtu vrcholů a hran) byla vynalezena tato metoda víceúrovňového dělení.

Víceúrovňové dělení má tři fáze: *zkracování (contraction)*, *interpolace (interpolation)*, *rozšíření (refinement)*.

Zkracování
Mějme graf $G = (V, E)$, který chceme „zkrátit“ na graf $G^{'} = (V^{'}, E^{'})$. Vybereme $V^{'}$jako maximální nezávislou množinu (viz obr. 1 - zeleně jsou vybarveny vrcholy spadající do nezávislé množiny daného grafu) vzhledem ke grafu G splňující tyto podmínky:

- $V^{'} \subseteq V$
- Žádný vrchol ve $V^{'}$ není propojen s hranami v E.
- Množina $V^{'}$je maximálně velká

Poté, co máme množinu $V^{'}$ můžeme sestrojit graf $G^{'}$. Myšlenka je taková, že vrcholy, které nejsou ve $V^{'}$ se přidávají do domén okolo vrcholů ve $V^{'}$, dokud nejsou všechny

vrcholy součástí domén obsahující vrchol z $V^{'}$. Domény D_i pro každý vrchol $i \in V^{'}$ obsahují jen jeho sousedy. $E^{'}$ tvoří hrany, které mají počátek a konec v jiné doméně. Nyní se dá jednoduše sestrojit „zkrácený“ graf $G^{'}$.

Interpolace
Druhý krok víceúrovňového algoritmu je interpolace. Cílem je dostat Fiedlerův vektor ze zkráceného grafu $G^{'}$, promítnout tento vektor na větší graf a použít tento vektor k poskytnutí dobrého přiblížení k dalšímu Fiedlerovu vektoru.

Mějme Fiedlerův vektor $f^{'} = (v_0, v_1, \ldots, v^{'}_{n-1}), n^{'} = \|V^{'}\|$ zkráceného grafu $G^{'}$. Sestrojíme rozšiřující vektor k většímu grafu $f^{''} = (v^{''}_0, v^{''}_1, \ldots v^{''}_{n-1}), n = \|V\|$ tak, že se použije jako přiblížení k Fiedlerovu vektoru pro originální graf G.

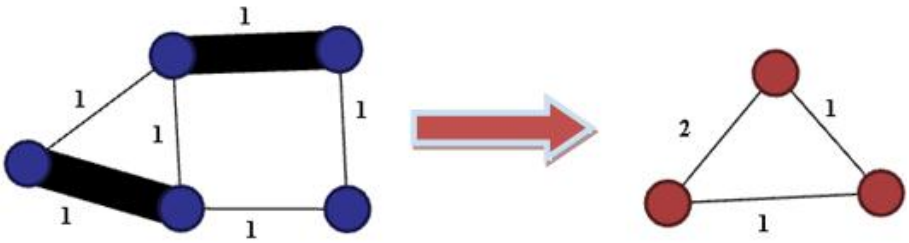
Rozšíření
Posledním krokem víceúrovňového dělení je rozšíření grafu $G^{'}$. Cílem je aplikovat a promítnout získáné hodnoty na původní graf G a pomocí těchto hodnot daný graf rozdělit.

4.4 Kombinace algoritmů

Podobným způsobem můžou být využity např. kombinace metody Maximálního párování (maximal matching), která slouží ke „zkracování“ grafu. Na toto rozdělení pak můžeme aplikovat některý z iteračních dělících algoritmů, jako je spektrální bisekce anebo Ford-Fulkersonův algoritmus (viz kapitola 4.6).

Maximální párování (Maximal matching)
Mějme souvislý, ohodnocený graf $G = (V, E)$. Maximální párování S v G je množina nepřilehlých hran, kde žádné dvě hrany nesdílejí společný vrchol. Párování S grafu G je maximální, jestliže má každá hrana v G společný vrchol s alespoň jednou hranou v S. Maximální párování obsahuje největší možný počet hran grafu G.

Příklad Maximálního párování je na obrázku 5. V levé části obrázku lze vidět graf s dvěma zvýrazněnými hranami. Tyto hrany tvoří S. V pravé části pak lze vidět výsledek maximálního párování.

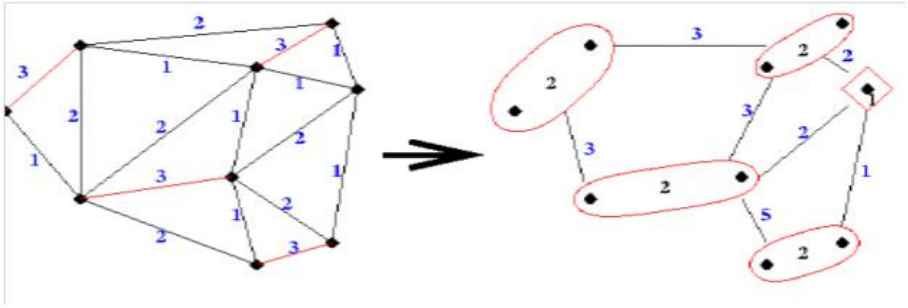


Obrázek 5: Ukázka Maximálního párování

Nyní, když je graf G zkrácený, můžeme aplikovat některý z bisekčních algoritmů.

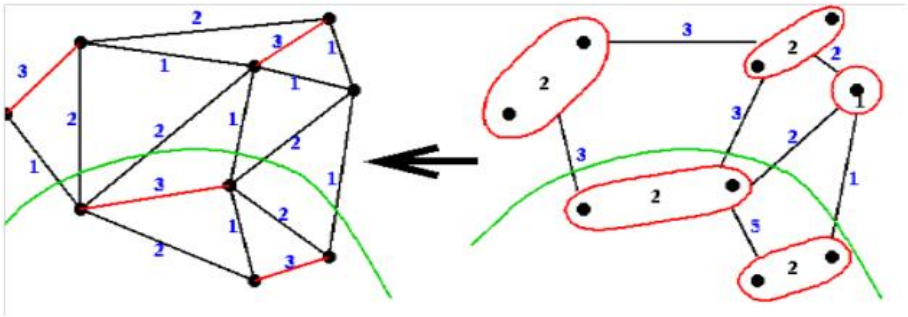
Příklad:

Mějme spojitý, ohodnocený graf $G = (V, E)$, který je na obrázku 6 v levé části. Aplikováním metody Maximální párování jsme dosáhli zkrácení grafu, který je v pravé části.



Obrázek 6: Ukázka zjednodušení grafu

Nyní, když máme zkrácený graf (méně vrcholů = menší časová náročnost) můžeme aplikovat např. Ford-Fulkersonův algoritmus pro hledání minimálního řezu. Na obrázku 7 lze v pravé části vidět nalezený minimální řez (označen zelenou barvou). Tento nalezený řez poté aplikujeme na původní graf a výsledek lze vidět v levé části tohoto obrázku.



Obrázek 7: Ukázka řezu zjednodušeným grafem

Tento postup se používá především na velmi velkých grafech, kdy zjednodušení grafu přináší velkou časovou úsporu.

4.5 Náhodná procházka (Random walk)

Náhodná procházka je algoritmus, u kterého se následující krok volí zcela náhodně. Jedná se tedy o nedeterministický algoritmus. Každý následující krok je závislý pouze na předchozí pozici a pravděpodobnostní funkci, která vymezuje následující směr. Tento vztah lze zapsat touto rovnicí:

$$X(t + \tau) = X(t) + \Phi(\tau),$$

kde $X(t)$ je cesta, kterou algoritmus doposud urazil a Φ je pravděpodobnostní funkce, která popisuje následující krok trvající čas τ.

Náhodná procházka začíná ve vrcholu V_0, který je pevně určen anebo vybrán dle pravděpodobnostního rozložení P_0. P_0 přiděluje každému vrcholu pravděpodobnostní hodnotu s jakou v něm bude algoritmus začínat. Algoritmus, začínající ve vrcholu V_0, po k-tém kroku se dostane do pozice V_k. Každý ze sousedů V_k má pravděpodobnost $\frac{1}{deg(v_k)}$, kde $deg(V_k)$ je stupeň vrcholu. Mějme matici M, která udává pravděpodobnost přechodu a je dána touto rovnicí:

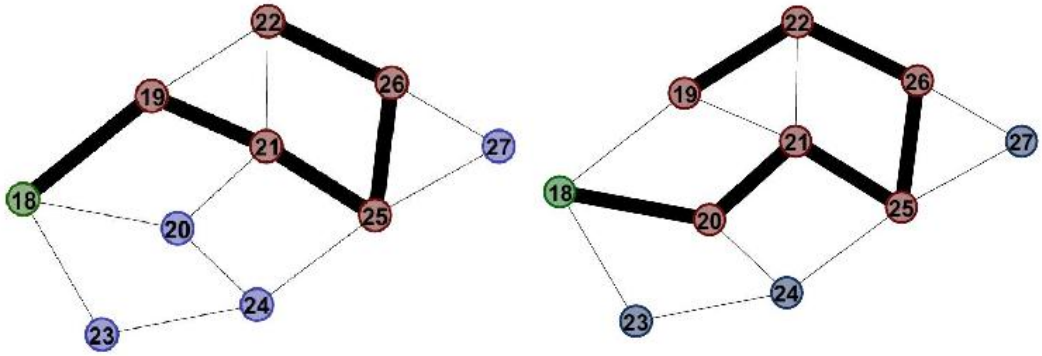
$$M_{ij} = \begin{cases} \frac{1}{deg(i)} & \text{jestli } \{i,j\} \in V \\ 0 & \text{jinak} \end{cases}$$

Postup, kterým se algoritmus řídí, udává tato rovnice:

$$P_{k+1} = M_T P_k,$$

kde P_k je vektor rozložení pravděpodobnosti přechodu po k krocích a T je počet kroků.

Příklad:



Obrázek 8: Ukázka Náhodné procházky

Na obrázku 8 lze vidět graf, kdy náhodná procházka má určen jako počáteční bod vrchol 18. Další krok, tedy výběr dalšího vrcholu, závisí na pravděpodobnosti sousedních vrcholů, která je pro tento krok vypočtena s ohledem na stupeň vrcholu (např. vrchol se stupněm 4 bude mít pravděpodobnost přechodu $\frac{1}{4}$). Po určení všech pravděpodobností sousedů se algoritmus náhodně rozhodne, který vrchol si vybere (avšak s přihlédnutím k pravděpodobnosti přechodu). Po přechodu na nový vrchol se pro jeho sousedy opět vypočítají pravděpodobnosti přechodu a celý cyklus se opakuje. Algoritmus pokračuje do té doby, než např. nebude mít žádný sousední vrchol, kterým ještě neprošel, k dispozici. Tak, jak je dokazuje obrázek 8, kde jsou ukázány dvě varianty průchodu grafem (nejsou

to zdaleka všechny možnosti). Omezení, kdy má tento algoritmus skončit je mnoho - dosažením některého vrcholu, množství vrcholů atd.

Existuje spousta různých variací Náhodné procházky. Pro představu, jak tento algoritmus pracuje, jsem vybral tento jednoduchý postup. Náhodná procházka objasňuje pozorované chování procesů v různých oblastech (ekologie, ekonomie, chemie, fyzika), a tak slouží jako základní model pro zaznamenané náhodné činnosti. Informace pro tuto podkapitolu byly čerpány z [18].

4.6 Ford-Fulkersonův algoritmus

Ford-Fulkersonův algoritmus je založen na jednoduché myšlence a to takové, že existuje-li cesta od zdroje ke spotřebiči taková, že se dá zvětšit tok na daných hranách, tak se zvýší na všech hranách této cesty tok o tuto hodnotu (kapacita zlepšující cesty). Tento postup se stále opakuje do té doby, než už nebude dále možno zvyšovat tok v síti, neboli nebude možno najít zlepšující cestu. Výsledkem tohoto algoritmu je nalezení maximálního toku sítě a množina určující minimální řez.

Věta (Ford-Fulkersonova věta o maximálním toku a minimálním řezu):
Velikost maximálního toku od zdroje z ke spotřebiči s je rovna kapacitě minimálního řezu oddělující zdroj a spotřebič.

Tato důležitá věta říká, v jakém vztahu jsou k sobě maximální tok a minimální řez v jedné konkrétní síti. Nikdy nemůže být maximální tok v síti vyšší, než je tok, který protéká minimálním řezem.

Nyní objasníme stručný návod, postup, krok po kroku, jak Ford - Fulkersonův algoritmus funguje:

Algoritmus 4: Popis postupu Ford-Fulkersnova algoritmu
Vstup: síť(G, z, s) s ohodnocenými hranami w a přípustným tokem f.
Výstup: maximální tok f, minimální řez sítí T.

1. Na vstupu je orientovaný graf G, u kterého je jednoznačně označen zdroj z, spotřebič s, celočíselné omezení hran (l a c) týkajících se toku v síti a celočíselný přípustný tok f.

2. Použije se značkovací procedura (která byla popsána již dříve - viz 2.2). Je-li označkován spotřebič s, bude se pokračovat následujícím krokem 3, není-li označkován, pokračuje se krokem 4.

3. Zjistí se kapacita d zlepšující cesty a změníme tok o tuto hodnotu. Dále se opět pokračuje krokem 1.

4. Neexistuje již zlepšovací cesta, tedy označíme A množinu označkovaných vrcholů. Výsledkem algoritmu je hodnota f, která představuje hodnotu maximálního toku a množina $W_{(A)}$, která určuje minimální řez.

Věta:

Ford-Fulkersonův algoritmus se po konečně mnoha krocích zastaví. Po zastavení je f maximálním tokem ze zdroje z do spotřebiče s a řez $W_{(A)}$ je minimálním řezem oddělujícím zdroj z a spotřebič s. Důkaz věty viz [4]

Časová složitost

Časová složitost tohoto algoritmu není závislá na velikosti vstupu, tedy na počtu hran a vrcholů, ale na kapacitách hran. V případě celočíselných kapacit hran je zaručeno, že se algoritmus po řadě kroků zastaví. V případě neceločíselných ohodnocení hran, může algoritmus být nekonečný.

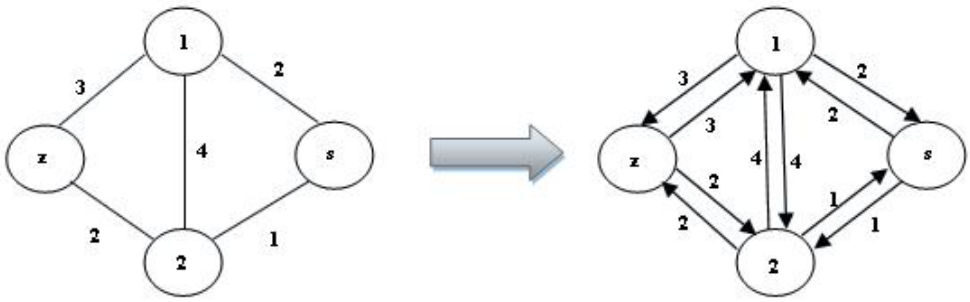
Pro algoritmus uvedený výše, za předpokladu celočíselných ohodnocení hran, platí časová složitost $O(|V| * |E|^2)$, $O(|V|^2 * |E|)$ nebo $O(|V|^3)$, kde V jsou vrcholy a E jsou hrany grafu.

Ford-Fulkersonův algoritmus se stal „základním stavebním kamenem“ pro mnoho dalších algoritmů, které jako základ používají „kostru“ tohoto algoritmu. Asi nejlepším příkladem je Edmondův - Karpův algoritmus, který se liší od Ford-Fulkersonova algoritmu pouze tím, že má určeno pořadí výběru zlepšujících cest, je-li v daném kroku algoritmu více takových cest na výběr.

Modifikovaný Ford-Fulkersonův algoritmus pro neorientovaný graf

Tento algoritmus pracuje na stejném principu jako Ford-Fulkersonův algoritmus pro orientovaný graf. Také hledá zlepšující cesty v grafu. Čím se liší, je to, že přidává dopřednou a zpětnou orientaci hranám neorientovaného grafu.

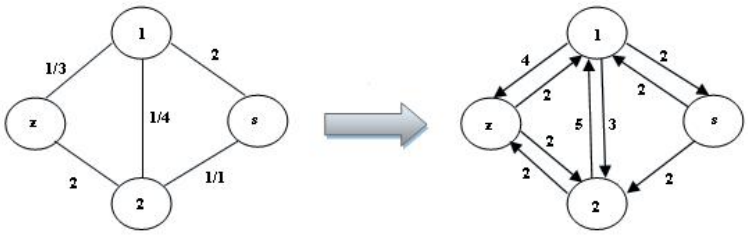
Příklad:



Obrázek 9: Transformace grafu

Mějme tento jednoduchý graf. V levé části obrázku 9 je zadán neorientovaný ohodnocený graf. V pravé části obrázku je jeho transformace na orientovaný ohodnocený graf. Z každé z hran mezi dvěma vrcholy se staly dvě orientované hrany - každá v opačném směru.

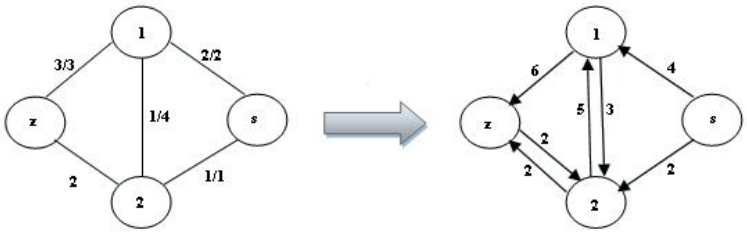
Krok 1:



Obrázek 10: Krok 1 v hledání minimálního řezu

V tomto kroku byla nalezena zlepšující cesta $(z, 1, 2, s)$, kdy maximální zlepšující tok je 1. V pravé části obrázku 10 je v dopředných hranách snížena kapacita o 1 (je již využita zlepšující cestou) a v hranách zpět jsou kapacity zvýšeny o 1 (Kirchhofovy zákony). Na hraně mezi vrcholy 2 a s došlo k maximálnímu využití dopředné hrany o kapacitě 1 a tedy lze již jen využít hranu vzad o kapacitě 2.

Krok 2:



Obrázek 11: Krok 2 v hledání minimálního řezu

V kroku 2 byla nalezena zlepšující cesta $(z, 1, s)$ a maximální tok zlepšující cesty je 2. Na obrázku 11 napravo vidíme, že byly vyčerpány dopředné cesty (hrany) mezi vrcholy z a 1 a mezi vrcholy 1 a s. Jelikož v dalším kroku již žádná zlepšující cesta nevede do spotřebiče s (hrany vedou jen ze spotřebiče, ne do něj), tak nyní se začne ze zdroje procházet graf. Vrcholy, které jsou dosažitelné ze zdroje z budou patřit do podgrafu A, zbylé vrcholy budou tvořit podgraf B. Hrany mezi těmito podgrafy tvoří minimální řez.

4.7 Kernighan - Linův algoritmus

Kernighan-Linův algoritmus je určen k následnému zlepšení minimálního řezu mezi dvěma podgrafy. Vstupem tohoto algoritmu je výsledek dělení grafu některého z dělících algoritmů (v případě této práce F-F a GGGP algoritmu). K-L algoritmus je založen na jednoduché myšlence. Mějme původní graf G, který obsahuje seznam hran E a seznam vrcholů V. Mějme také dva různé podgrafy A a B. Algoritmus přiřadí každému vrcholu ohodnocení D_A, D_B dle váhy hran, které z něj vycházejí.

Mějme vnitřní hodnotu I_a, která udává součet vnitřních váh hran v podgrafu A u daného vrcholu:

$$I_a = \sum_{v \in A} c_(av)$$

Mějme vnější hodnotu E_a, která udává součet váh hran směřujících z vrcholu v podgrafu A do některého z vrcholů v podgrafu B:

$$E_a = \sum_{v \in B} c_(av)$$

Mějme rovnici $D_a = E_a - I_a$, která udává rozdíl mezi vnější a vnitřní hodnotou váh hran u daného vrcholu. Dalším krokem, který algoritmus provede je výpočet cen (váh) cest ze všech vrcholů podgrafu A do všech vrcholů podgrafu B pomocí této rovnice:

$$g_{ab} = D_a + D_b - 2c_{ab},$$

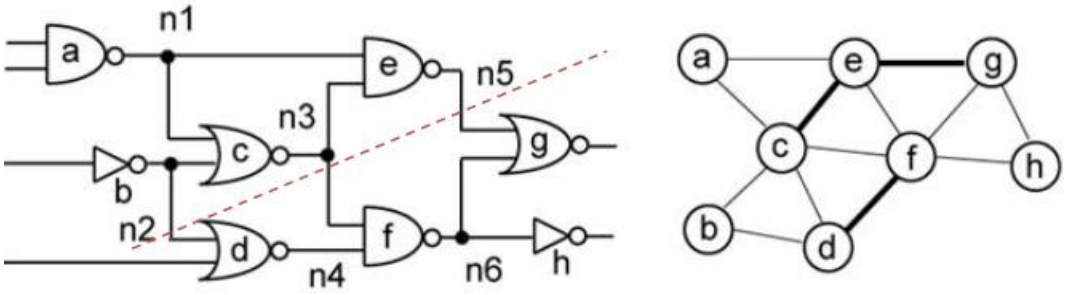
kde c_{ab} udává hodnotu váhy hrany, jestliže jsou vrcholy v_a a v_b propojeny hranou. V opačném případě $c_{ab} = 0$.

V následujícím kroku algoritmu se vybere taková cesta g_{ab} , která má nejvyšší váhu. Tato cesta (dvojice vrcholů) se zamění (vrchol z podgrafu A putuje do podgrafu B a naopak, vrchol z podgrafu B putuje do podgrafu A) a přepočítají se hodnoty D_a. Následuje opětovný postup při hledání dvojice vrcholů o nejvyšší váze cesty, avšak bez vrcholů, které byly v předchozích krocích vyměněny. Díky tomu je zajištěno, že algoritmus po k krocích skončí se zaměňováním vrcholů. Nyní provede hledání maximální hodnoty dvojice vrcholů nebo součet těchto dvojic, které skutečně v daném grafu zamění.

Algoritmus 5: Kernighan-Linův algoritmus
Vstup: *síť (G, z, s) s ohodnocenými hranami w.*
Výstup: *seznam vrcholů a hran podgrafů A, B z původního grafu G.*

Krok 1: V = 2n; A, B jsou podgrafy grafu G, kde platí: $A \cap B = \emptyset$
Krok 2: Výpočet D_v pro všechny $v \in V$; $A^{'} = A, B^{'} = B$
Krok 3: Výběr prvků $a_i \in A^{'}, b_i \in B^{'}$, které mají maximální hodnotu
Výpočet všech $g_i = D_{a_i} + D_{b_i} - 2c_{a_i b_i}$
$A^{'} = A^{'} - \{a_i\}, B^{'} = B^{'} - \{b_i\}$;
Krok 4: Jestliže $A^{'}$ a $B^{'}$ jsou prázdné, pak Goto: Krok 5)
jestliže nejsou prázdné: přepočet D hodnot pro $A^{'} \cup B^{'}$ a Goto: Krok 3
Krok 5: přehození vrcholů podgrafů A, B vzhledem k výpočtu
$G = \sum_{i=1}^{k} g_(i)$;
Jestliže $G > 0$, pak
přesunout $X = \{a_i, ..., a_k\}$do B;
přesunout $Y = \{b_i, ..., b_k\}$do A;
Goto Krok 2;
Jestliže $G < 0$, pak
konec

Příklad:



Obrázek 12: Příklad postupu Ford-Fulkersonova algoritmu

V levé části obrázku 12 lze vidět schéma elektrického zapojení a v pravé části vidíme totéž schéma jen překreslené na graf, na který se nyní aplikuje Kernighan - Linův algoritmus. Předpokládejme, že váhy hran jsou všechny rovny 1 a graf jsme rozdělili na tyto dva podgrafy: $A = \{a, b, d, e\}, B = \{c, f, g, h\}$

Krok 1:

Pár	$E_x - I_x$	$E_y - I_y$	$c(x, y)$	g_{xy}
(a, c)	$0.5 - 0.5$	$2.5 - 0.5$	0.5	1
(a, f)	$0.5 - 0.5$	$1.5 - 1.5$	0	0
(a, g)	$0.5 - 0.5$	$1 - 1$	0	0
(a, h)	$0.5 - 0.5$	$0 - 1$	0	-1
(b, c)	$0.5 - 0.5$	$2.5 - 0.5$	0.5	1
(b, f)	$0.5 - 0.5$	$1.5 - 1.5$	0	0
(b, g)	$0.5 - 0.5$	$1 - 1$	0	0
(b, h)	$0.5 - 0.5$	$0 - 1$	0	-1
(d, c)	$1.5 - 0.5$	$2.5 - 0.5$	0.5	2
(d, f)	$1.5 - 0.5$	$1.5 - 1.5$	1	-1
(d, g)	$1.5 - 0.5$	$1 - 1$	0	1
(d, h)	$1.5 - 0.5$	$0 - 1$	0	0
(e, c)	$2.5 - 0.5$	$2.5 - 0.5$	1	2
(e, f)	$2.5 - 0.5$	$1.5 - 1.5$	$0,5$	1
(e, g)	$2.5 - 0.5$	$1 - 1$	1	0
(e, h)	$2.5 - 0.5$	$0 - 1$	0	1

Tabulka 1: Kernighan - Lin: krok 1

V tabulce 1 jsou znázorněny v prvním sloupci páry (vždy každý z podgrafu A s každým z podgrafu B). V druhém a třetím sloupci je výpočet hodnot D_x, D_y, které udávají poměr mezi externími a interními hodnoty hran daného vrcholu. Čtvrtý sloupec udává váhu hrany (jestliže existuje) mezi danými vrcholy. Poslední sloupec udává výslednou

váhu cesty daného páru (z vrcholu x do vrcholu y). Z tabulky vyplývá, že nejvyšší hodnota a tedy nejvhodnějším kandidátem na výměnu je pár(d, c), kdy výsledná hodnota je 2.

Nyní se celý proces opakuje, jen s tou změnou, že již se nebudou brát v potaz zaměněné vrcholy (d, c).

Krok2:

Pár	$E_x - I_x$	$E_y - I_y$	$c(x, y)$	g_{xy}
(a, f)	$0 - 1$	$1 - 2$	0	-2
(a, g)	$0 - 1$	$1 - 1$	0	-1
(a, h)	$0 - 1$	$0 - 1$	0	-2
(b, f)	$0.5 - 0.5$	$1 - 2$	0	-1
(b, g)	$0.5 - 0.5$	$1 - 1$	0	0
(b, h)	$0.5 - 0.5$	$0 - 1$	0	-1
(e, f)	$1.5 - 1.5$	$1 - 2$	$0,5$	-2
(e, g)	$1.5 - 1.5$	$1 - 1$	1	-2
(e, h)	$1.5 - 1.5$	$0 - 1$	0	-1

Tabulka 2: Kernighan - Lin: krok 2

Z této tabulky 2 vyplývá, že nejvhodnějším kandidátem na výměnu je pár (b, g), jejichž hodnota je 0. Tato hodnota, je-li záporná, udává, že záměnou by se budoucí rozdělení zhoršilo (mělo by vyšší váhu). Je-li však hodnota kladná, znamená to, že záměnou se budoucí rozdělení zlepší (bude mít menší váhu mezi podgrafy). Hodnota 0 znamená, že záměnou těchto vrcholů se budoucí rozdělení nezlepší, avšak ani nezhorší. Zůstane pořád na stejné hodnotě.
Algoritmus tedy vybere pár (b, g) a opět pokračuje následujícím výpočtem.

Krok 3:

Pár	$E_x - I_x$	$E_y - I_y$	$c(x, y)$	g_{xy}
(a, f)	$0 - 1$	$1.5 - 1.5$	0	-1
(a, h)	$0 - 1$	$0.5 - 0.5$	0	-1
(e, f)	$0.5 - 2.5$	$1.5 - 1.5$	0.5	-3
(e, h)	$0.5 - 2.5$	$0.5 - 0.5$	0	-2

Tabulka 3: Kernighan - Lin: krok 3

Zde výsledek vyšel záporný, tedy již víme, že touto záměnou by se budoucí rozdělení zhoršilo, avšak algoritmus musí doběhnout až do konce. Párům (a, f) a (a, h) vyšly stejné hodnoty a tak algoritmus náhodně vybere jeden z nich (nyní např. pár (a, f)).
Nyní již zbývají pouze 2 vrcholy, každý v jednom z podgrafů.

Krok 4:

Pár	$E_x - I_x$	$E_y - I_y$	$c(x, y)$	g_{xy}
(e, h)	$0.5 - 2.5$	$1.5 - 0.5$	0	-1

Tabulka 4: Kernighan - Lin: krok 4

Když je algoritmus u konce a nezbývají již žádné vrcholy k záměně, vypočte se optimální řešení pro dané rozdělení. Algoritmus zpětně vypočte hodnoty záměn takto:
První záměna páru $(d, c) = 2$.
Druhá záměna páru $(b, g) = 0$, ale k této hodnotě algoritmus musí přičíst hodnotu první záměny, jelikož touto záměnou byla ovlivněna, tedy výsledná hodnota je 2 (2+0).
Výsledky jsou uvedeny v tabulce (viz tabulka 5) níže.

Krok	**Pár**	$g_{xy}(krok)$	$\sum g_{xy}(krok)$	**Velikost řezu**
1	(d, c)	2	2	3
2	(b, g)	0	2	3
3	(a, f)	-1	1	4
4	(e, h)	-1	0	3

Tabulka 5: Kernighan - Lin: Výsledek

Z tabulky 5 vyplývá, že ideální záměnou v podgrafech by bylo prohodit vrchol d s vrcholem c a nebo vrholy d, b s vrcholy b, g. Jelikož u páru (b, g) vyšel výsledek 0, tedy prohozením tohoto páru by se nic nestalo, prohodíme pouze pár (d, c).
Výsledný řez je tedy: $A = \{a, b, c, e\}, B = \{d, f, g, h\}$ a je zobrazen na obrázku 12 červenou přerušovanou čárou.[20]

Časová složitost

Výpočetní složitost výpočtu D_a hodnot má složitost $O(n^2)$, pro každý vrchol $O(n)$. Po záměně vrcholů se musí aktualizovat D_a hodnoty, přičemž složitost tohoto kroku je $O(2n - 2i)$ po záměně páru (a_i, b_i). Celková složitost aktualizací D_a hodnot je:

$$\sum_{i=1}^{n} (2n - 2i) = O(n^2)$$

Procedura na výběr páru v grafu je nejdražším krokem algoritmu. Výběr párů má časovou složitost $(n - i + 1)^2$. Celková složitost je tedy $O(n^3)$. [20]

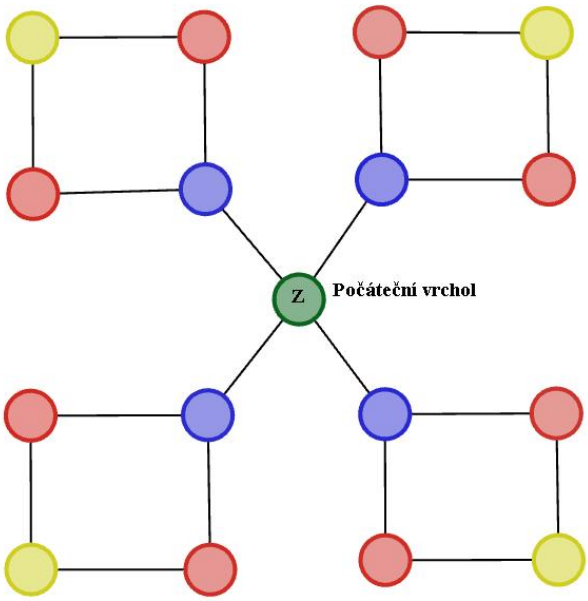
4.8 Greedy graph growing partitioning algoritmus (GGGP)

Jedná se o bisekční algoritmus (čerpáno z [21]), který lze využit při víceúrovňovém dělení grafů. Výsledek dělení grafu z velké části závisí na vybraném počátečním vrcholu grafu. Jelikož časová složitost tohoto algoritmu je $O(mn$, kde m je počet vrcholů a n je počet hran), může se tento algoritmus volat několikrát (např. 5-krát), pokaždé s jiným zvoleným

počátečním vrcholem a výsledek s nejmenším minimálním řezem se vybere jako nejlepší dělení daného grafu.

Algoritmus je založen na postupném přidávání hran do seznamu hran E_1 budoucího podgrafu G_1, spojujících sousedné vrcholy s počátečním vrcholem. Na začátku algoritmu je počáteční vrchol vložen do seznamu vrcholů V_1 budoucího podgrafu G_1. Posléze se naleznou jeho sousedé a přidají se rovněž do V_1. Naleznou se také hrany spojující tyto vrcholy z V_1. Jestliže součet váh hran v E_1 je větší, než součet váh hran původního grafu, algoritmus končí a V_1 a E_1 tvoří podgraf G_1 a zbylé vrcholy a hrany mezi nimi tvoří podgraf G_2. Jestliže je váha hran menší, opakuje se stejný postup, kdy ke každému vrcholu z V_1 jsou nalezeni sousedé, ti posléze přidání do V_1 a jsou také nalezeny hrany mezi vrcholy V_1, přidány do E_1 a algoritmus opět vyhodnotí, zda je součet váh hran z E_1 větší než celkový součet váh hran původního grafu.

Tento postup byl využit při implementaci tohoto algoritmu v testovací aplikaci. Aplikace METIS má tento algoritmus naimplementován tak, že místo váh hran přidává vrcholy do V_1 tak dlouho, než počet vrcholů je větší nebo roven polovině celkového počtu vrcholů v grafu G. Porovnání výsledků z těchto dvou modifikací GGGP algoritmu je uvedeno v kapitole 6.



Obrázek 13: Příklad postupu GGGP algoritmu

Na obrázku lze vidět, jak postupně graf přidává jednotlivé uzly do V_1, přičemž počáteční vrchol je zelený.

5 Způsoby měření kvality dělení grafu

U grafových algoritmů pro dělení grafů není předem definováno, jak mají výsledné komunity vypadat, kde a do jaké skupiny mají dané vrcholy patřit a tedy je velmi důležité sledovat jejich kvalitu dělení. Hlavní nevýhodou měření kvality komunit je ten fakt, že nelze vybrat jeden reprezentující vrchol z dané komunity a „změřit" jeho kvalitu. Informace byly čerpány z těchto zdrojů [5, 6, 7].

Existují různé algoritmy pro dělení grafu, které můžeme zařadit do některých z těchto čtyř skupin:

- *Partitional clustering* - tyto algoritmy rozkládají přímo daná data do disjunktních komunit. Snaží se určit celočíselný počet oddílů, které optimalizují podle daného kritéria. Většinou se jedná o dělení do k komunit, kdy číslo k je předem definované. Dělení do několika komunit se docílí pomocí iterace.
- *Hiearchical clustering* - zde spadají algoritmy, které tvoří komunity rekurzivně. Komunity se vytváří sloučením více menších komunit do jedné větší nebo naopak z větší komunity vzniknou komunity menší.
- *Density - based clustering* - komunity jsou pomocí těchto algoritmů identifikovány jako oblasti s vyšší hustotou, než je hustota ve zbývající části datového souboru.
- *Grid - based clustering* - tyto typy algoritmů jsou především navrženy pro prostorové dolování dat.

Některé indexy kvality, které budou použity při určování kvality dělení grafů při experimentech na výsledných grafech (viz kapitola 6), si zde uvedeme.

5.1 Modularita

Modularita je jedna ze základních hodnoticích prvků při měření kvality shlukování. Slouží především k detekci komunit v sociálních sítích. Základním principem je měření síly rozdělení sítě do komunit. Sítě s vysokou modularitou mají obvykle husté spojení mezi sebou v rámci komunity a nízkou hustotu spojení mezi vrcholy ostatních komunit. Modularita je zlomek hran, které spadají do dané skupiny, mínus očekávané zlomky, jestliže okraje byly rozděleny náhodně. Nabývá hodnot $[-1, 2.1)$, kdy kladná hodnota znamená, že počet hran v komunitách oproti očekavání je vyšší.
Existuje mnoho metod pro určení modularity. Nejběžnější způsob, jak určit modularitu je náhodné zvolení okrajů tak, aby se zachovaly stupně každého vrcholu. Mějme graf s n uzly a m hranami. Tento graf se rozdělí do dvou skupin s atributem členství s. Jestliže vrchol v_i bude patřit do první komunity, tak $s_i = 1$, jestliže do druhé, pak$s_i = -1$. Mějme matici sousednosti, kde $A_{ij} = 0$, jestliže mezi vrcholy i, j není žádná hrana. Existuje-li mezi vrcholy i, j hrana, pak $A_{ij} = 1$. Pro neorientovaný graf platí $A_{ij} = A_{ji}$.

Modularita Q je pak definována jako podíl hran, které spadají do skupiny 1 nebo 2, mínus očekávaný počet hran v rámci skupiny 1 a 2 pro rozdělení náhodného grafu o stejném stupni vrcholů.

Očekávaný počet hran se vypočte pomocí konceptu konfigurace modelů. Konfigurační model je náhodná realizace konkrétní sítě. Mějme síť s n uzly, kde každý uzel i má stupeň k_i. Konfigurační model rozdělí každou hranu na dvě poloviny (tzv. „útržky“) a náhodně je pospojuje s ostatními hranami. Tak zůstane úroveň uzlu stejná a vznikne nový náhodný graf se stejným stupněm vrcholů.

Celkový počet „útržků“ se určí touto rovnicí:

$$l_n = \sum_{i}^{n} k_i = 2m$$

Výpočet modularity se pak provede pomocí této rovnice:

$$Q = \frac{1}{2m} \sum_{ij} [A_{ij} - \frac{k_i * kj}{2m}] \frac{s_i s_j + 1}{2}$$

Tento postup je vhodný pouze pro určování modularity při jednom rozdělení komunity do 2 menších komunit. Je-li potřeba dále rozdělit i tyto sub komunity na další menší komunity a určit celkovou modularitu pak rovnice vypadá takto:

$$Q = \sum_{ij} [\frac{A_{ij}}{2m} - \frac{k_i * kj}{(2m) * (2m)}] \sigma(c_i, c_j) = \sum_{ij}^{c} (e_{ii} - a_i^2)$$

kde e_{ii} je podíl hran s oběma koncovými vrcholy ve stejné komunitě i:

$$e_{ii} = \sum_{j} \frac{A_{ij}}{2m} \sigma(c_i, c_j)$$

a a_i je zlomek hran s alespoň jedním vrcholem v komunitě i:

$$a_i = \frac{k_i}{2m} = \sum_{j} a_{ij}$$

Měření kvality sítě pomocí modularity má jedno velké omezení a to určitou mez, která detekuje komunity a která je přímo úměrná velikosti testované sítě. Při měření modularity velkých sítí může dojit k přehlédnutí menších komunit.

5.2 Surprise

Surprise se používá k určování kvality rozdělení sítí, především v sociálních sítích. Jedná se o novější formulaci modularity, „vylepšenou“ verzi, kdy problém s detekcí menších komunit je zde vyřešen.

Mějme danou skupinu rozdělenou do komunit. Surprise porovnává počet vnějších a vnitřních vazeb mezi komunitami v této skupině s předpokládaným počtem vazeb v náhodné síti se stejným rozdělením uzlů do komunit. Surprise S nabývá hodnot $[0, \infty)$. Surprise S je definován takto:

$$S = -log \sum_{j=p}^{min(M,n)} \frac{\binom{M}{j}\binom{F-M}{n-j}}{\binom{F}{n}},$$

kde n je skutečný počet vazeb v síti a p je skutečný počet vnitro-komunitních vazeb tohoto oddílu. F je maximální možný počet vazeb v síti pro počet k vrcholů:

$$F = \frac{k(k-1)}{2}$$

M je maximální možný počet vnitro-komunitních vazeb. Jestliže K je počet komunit, pak:

$$M = \sum_{i=1}^{K} \frac{k_i(k_i-1)}{2}$$

5.3 Indexy založené na měření vnějšího/vnitřního propojení vrcholů uvnitř a mezi komunitami

Většinou se kvalita rozdělení grafu měří za pomoci hran, které spojují dva různé podgrafy. Obecně je dáno, čím méně hran propojuje dva různé podgrafy, tím je rozdělení těchto podgrafů lepší.

Cut index (nebo také MaxMinCut)
Cut index udává poměr mezi počtem hran uvnitř podgrafu a počtem vnějších hran, vycházejících z tohoto podgrafu. Nebere však v potaz velikost (počet vrcholů) podgrafu.

Je dán rovnicí:

$$MaxMinCut = \sum_{i=1}^{K} \frac{E_i'}{E_i},$$

kde E_i' je počet hran mezi K_i-tým podgrafem a jiným podgrafem a E_i je počet hran v K_i-tým podgrafu.

Pokrytí (coverage)
Pokrytí grafového dělení je dáno podílem sumy vnitřních hran (E_i) podgrafů k celkovému počtu hran (E) celého grafu.

Pokrytí je dáno rovnicí:

$$Cov(C) = \frac{\sum_{i=1}^{K} E_i}{E},$$

kde K je počet podgrafů.

6 Vlastní implementace a experimenty

V této kapitole je prezentována testovací aplikace Graph_Partitioning - popis tříd této aplikace. Dále jsou zde představeny naměřené výsledky testů prováděných nad různě velkými grafy. Na konci této kapitoly jsou vyhodnoceny výsledky testů a určena vhodnost jednotlivých algoritmů dle výsledků testů.

6.1 Popis aplikace

Aplikace je vytvořena jako konzolová aplikace a je naimplementována nad platformou .NET v jazyce C# . Skládá se z několika tříd:

- Třída *Node* - tato třída reprezentuje vrchol daného grafu. Jejími atributy jsou (podle formátu GDF souboru) ID vrcholu, název vrcholu, rozměr vrcholu x,y a také RGB atributy pro definování barvy vrcholu.
- Třída *Edge* - tato třída reprezentuje hranu daného grafu. Je dána atributy (dle formátu GDF): ID hrany, název hrany, vrchol 1, vrchol 2, váha hrany, maximální tok vpřed a vzad a RGB atributy pro definování barvy hrany.
- Z těchto dvou tříd se skládá třída *Graph*. Tato třída je definována jako seznam hran (Edge) a vrcholů (Node).
- Třída *Modularity* slouží k otestování kvality dělení grafu daným algoritmem. Třída Modularity měří modularitu děleného grafu.
- Třída *MaxMinCut* také slouží k měření kvality dělení grafu. Třída MaxMinCut měří MaxMinCut index děleného grafu.
- Dále se zde nachází třída *ReadGDF*, která slouží pro převod z GDF souboru do testovací aplikace (konkrétně do instance třídy Graph). Tato třída umožňuje také opačný směr převodu dat - z instance třídy Graph do souboru GDF.
- Další třídou, která umožňuje převod souboru z/do instance třídy Graph je třída *ReadFromMetisFile*. Tato třída převádí formát souboru (GRAPH) z aplikace METIS do instance třídy Graph a naopak. Také výsledek z aplikace METIS (jiný formát souboru) bylo potřeba převést do instance třídy Graph a následně do souboru GDF pro jeho vizualizaci.
- Třída *Ford_Fulkerson* realizuje vybraný Ford-Fulkersonův algoritmus (viz 4.6). Tento algoritmus byl vybrán záměrně proto, že není součástí aplikace Metis a bude srovnávána právě s výsledky z této aplikace.
- Třída *GreedyGraphGrowingPartitioning* realizuje druhý zvolený algoritmus GGGP (Greedy Graph Growing Parititoning - viz 4.8). Tento algoritmus obsahuje jak tato testovací aplikace, tak i aplikace Metis. Tyto dva výsledky budou na konci porovnávány s výsledky Ford-Fulkersonova algoritmu.

- Třída *KernighanLin* realizuje vybraný Kernighan-Linův algoritmus (viz 4.7). Tato třída se „volá“ po rozdělení grafu na víc části.

Ostatní třídy slouží k určení zdroje a spotřebiče:

- Třída *BiggestDegreeNodeFromNodes* vrátí dva vrcholy z grafu s nejvyšším stupněm vrcholu.
- Třída *FarthestPathFromNode* naopak vrátí dva vrcholy od sebe nejvzdálenější.
- Třída *RepairsToContinuosGraph* slouží k odstranění nesouvislosti grafu po aplikaci GGGP algoritmu.
- Poslední třídou je testovací třída *Test_Algorithms*, v které jsou prováděny a vyhodnocovány testy jednotlivých algoritmů.

6.2 Cíle a předpoklady testů

Dříve, než se uskuteční testování, je třeba si určit cíle a předpokládané výsledky těchto testů. Jelikož každý z použitých algoritmů má svá specifická omezení jak na kvalitu, tak na čas a byly detailně rozebírány v předešlých kapitolách, lze některé výsledky testů s jistou pravděpodobnosti předpovědět.

Při testování Ford-Fulkersonova algoritmu se očekává výsledné rozdělení grafů s velmi dobrými výsledky - vysoká modularita a nízký MaxMinCut index. Dalším předpokladem je jeho vysoká časová náročnost u velkých grafů (více než 1000 vrcholů), jelikož jeho časová náročnost je $O(n^3)$. Posledním předpokládaným výsledkem je ta skutečnost, že po aplikování Kernighan-Linova algoritmu, na rozdělení pomocí Ford-Fulkersonova algoritmu, se modularita a MinMaxCut index téměř nezmění, ne-li vůbec.

Při testování GGGP na grafech je předpoklad úplně opačný. Časová náročnost tohoto algoritmu bude mnohen nižší, než u Ford-Fulkersonova algoritmu, vzhledem k jeho časové náročnosti $O(mn$, kde m je počet vrcholů a n je počet hran). Také po aplikování Kernighan-Linova algoritmu se bude jeho modularita a MaxMinCut index výrazněji lišit od původního rozdělení, jelikož tento algoritmus není založen na hledání minimálního řezu, ale pouze rozpůlí daný graf na dvě části podle váh hran.

U výsledků testů z aplikace METIS se očekávají nejkvalitnější a nejrychlejší výsledky. Časová náročnost bude velmi nízká vzhledem k optimalizaci kódu, který v testovací aplikaci není na takové úrovni. Také bude časový rozdíl velmi patrný u velkých grafů. Díky aplikování algoritmu na zkrácení grafu a následnému aplikování GGGP na zkrácený graf, bude časová náročnost výrazně nižší. Otázkou však zůstává, jestliže výsledné rozdělení bude mít lepší modularitu a MaxMinCut index, než např. rozdělení pomocí Ford-Fulkersonova algoritmu. Jelikož aplikace METIS zahrnuje zkracování grafů, následnou aplikaci vybraného algoritmu (GGGP), poté rozšíření a následně aplikaci Kernighan-Linova algoritmu na rozdělený graf, nebude možno změřit kvalitu rozdělení grafu jen pomocí GGGP.

Při testování budou použity tři různé typy grafů. Edison - je neorientovaný ohodnocený graf, který mi byl přidělen mým vedoucím a zachycuje syntetické vztahy mezi

studenty. Metis_mesh - jedná se o mesh, která byla součástí aplikace METIS, určená k testování. Graf100 - jedná se o neorientovaný, ohodnocený graf se 109 vrcholy a byl mnou uměle vytvořen za účelem testování algoritmů na malých grafech.

6.3 Výsledky testů

Nyní budou prezentovány jednotlivé testy. Každý test probíhá na třech různých grafech - velký graf (Edison) s přibližně 2 000 vrcholy, malý graf (Graf100) s přibližně 100 vrcholy a mesh(Metis_mesh) s přibližně 7000 vrcholy.

Výsledky všech provedených testů:

1. Testování Ford-Fulkersonova algoritmu na grafech s nejvzdálenějším zdrojem od spotřebiče.

Název grafu	Počet vrcholů	Počet hran	Čas dělení grafu [00:00:000]	Počet vrcholů podgrafu G1	Počet hran podgrafu G1
Edison	2133	51285	-	-	-
Graf100	109	154	00:00:047	64	87
Metis_mesh	7434	22302	03:18:281	1	0

Počet vrcholů podgrafu G2	Počet hran podgrafu G2	Minimální řez po F-F	MaxMin-Cut po F-F	Modularita po F-F	Počet změn po K-L
-	2133	51285	-	-	-
45	63	4	0,0242	0,49965	0
7433	22299	3	0	0	0

Minimální řez po K-L	MaxMin-Cut po K-L	Modularita po K-L
-	-	-
4	0,0242	0,49965
3	0	0

Tabulka 6: Výsledek testů F-F s 2 nejvzdálenějšími vrcholy grafu jako zdroj a spotřebič

V tabulce 6 vidíme výsledky testů dělení grafů pomocí F-F algoritmu. U tohoto dělení byly určeny dva počáteční parametry - dva vrcholy (zdroj a spotřebič) jako dva nejvzdálenější od sebe, čímž se dosáhlo velmi dobrých výsledků. Na grafu Edison však nebylo možno naměřit jakékoliv hodnoty, jelikož tento graf má 2133 vrcholů a tento algoritmus má časovou složitost $O(n^3)$. Ukázalo se tedy, že F-F algoritmus není vhodný pro dělení grafů obsahujících více, než 2000 vrcholů. Což ovšem není až tak pravda, jelikož také

hodně záleží, z hlediska časového, na ohodnocení váh hran daného grafu. Například u grafu Metis_mesh, který reprezentuje mesh, jsou všechny hrany ohodnoceny váhou 1, a tedy při hledání zlepšujících cest většinou algoritmus prochází touto hranou pouze jednou, pak je již kapacita této hrany vyčerpaná. Z tohoto důvodu, také šlo z hlediska časového rozdělit Metis_mesh, který obsahuje přes 7000 vrcholů (viz 6). Výsledek tohoto dělení na daném grafu dopadl tak, že algoritmus oddělil zdroj od zbytku meshe. Tento výsledek se dal očekávat vzhledem k hustotě hran v tomto grafu. Důležitý je však výsledný čas, který je 3min. a 18s. Vzhledem ke grafu Edison, který obsahuje třikrát méně vrcholů a jehož čas dělení se odhaduje v řádu hodin, je tento výsledek z hlediska časového výborný. Avšak tento čas je ovlivněn také strukturou grafu, kdy se do podgrafu G1 dostal jen jediný vrchol a tedy algoritmus rychle skončil. Posledním testovaným grafem v tomto testu byl Graf100. Tento graf obsahuje 109 vrcholů. Výsledek tohoto dělení lze vidět na obrázku 16. Čas dělení byl velmi rychlý 0,047s. Kvalita výsledného grafu je velmi vysoká, jelikož po aplikaci K-L algoritmu již nedošlo k přehození jakéhokoliv vrcholu a tedy se ani minimální řez nezměnil. Modularita je velmi vysoká a MaxMinCut index velmi nízký.

2. Testování Ford-Fulkersonova algoritmu na grafech s dvěma vrcholy o nejvyšším stupni - jako zdroj a spotřebič.

Název grafu	Počet vrcholů	Počet hran	Čas dělení grafu [00:00:000]	Počet vrcholů podgrafu G1	Počet hran podgrafu G1
Edison	2133	51285	-	-	-
Graf100	109	154	00:00:125	43	61
Metis_mesh	7434	22302	08:21:375	7433	22291

Počet vrcholů podgrafu G2	Počet hran podgrafu G2	Minimální řez po F-F	MaxMin-Cut po F-F	Modularita po F-F	Počet změn po K-L
-	2133	51285	-	-	-
66	89	4	0,0242	0,4965335	0
1	0	11	0	0	0

Minimální řez po K-L	MaxMin-Cut po K-L	Modularita po K-L
-	-	-
4	0,0242	0,4965335
11	0	0

Tabulka 7: Výsledek testů F-F s 2 vrcholy o nejvyšších stupních v grafu

V tabulce 7 vidíme naměřené hodnoty. Tento test se od toho předchozího liší pouze v nastavení počátečního zdroje a spotřebiče pro F-F algoritmus. Nyní se našly dva vrcholy o nejvyšším stupni (viz 17). Graf Edison nemohl být vzhledem k časové náročnosti F-F algoritmu proveden. Mesh s názvem Metis_mesh byla rozdělena za více než 8min. Taky zde došlo k odtržení jediného vrcholu od zbytku grafu. Důvodem toho rozdělení je, že druhý vrchol (spotřebič) byl na „okraji" sítě a F-F algoritmus ho tedy oddělil od zbytku sítě. Z tabulky lze vyčíst, že tento vrchol měl 11 sousedních vrcholů (minimální řez je 11). Je tedy zřejmé, že F-F algoritmus se pro dělení mesh nehodí. Výsledek grafu Graf100 (viz obrázek 17) dopadl stejně, jak u předchozího testu, jen je časově pomalejší.

3. Testování GGGP algoritmu s nejvzdálenějšími vrcholy mezi sebou.

Název grafu	**Počet vrcholů**	**Počet hran**	**Čas dělení grafu [00:00:000]**	**Počet vrcholů podgrafu G1**	**Počet hran podgrafu G1**
Edison	2133	51285	02:25:187	1825	43665
Graf100	109	154	00:00:016	44	56
Graf100_upd	109	154	00:00:016	48	63
Metis_mesh	7434	22302	00:50:156	5723	15626

Počet vrcholů podgrafu G2	**Počet hran podgrafu G2**	**Minimální řez po F-F**	**MaxMin-Cut po GGGP**	**Modularita po GGGP**	**Počet změn po K-L**	**Minimální řez po K-L**	**MaxMin-Cut po K-L**	**Modularita po K-L**
308	4995	309010	0,303	0,49999	-	-	-	-
65	84	20	0,1221	0,49688	44	34	0,0242	0,49739
61	82	15	0,0861	0,49763	2	8	0,0518	0,4977
1711	1167	5509	3,4972	0,499628	-	-	-	-

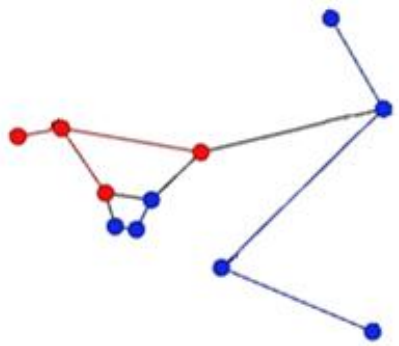
Tabulka 8: Výsledek testů GGGP s 2 nejvzdálenějšími vrcholy v grafu

Pomocí tohoto algoritmu s časovou složitostí $O(mn$, kde m je počet vrcholů a n je počet hran) lze rozdělit i větší grafy, než u F-F algoritmu. Z tabulky 8 lze vyčíst, že graf Edison byl rozdělen na dvě části za 2:25 min. Výsledný graf lze vidět na obrázku 18. Rozdělení proběhlo s tímto výsledkem, jelikož tento hladový algoritmus postupně

ke svému počátečnímu vrcholu přidává další vrcholy a počítá váhu hran mezi těmito vrcholy. Jakmile je váha hran mezi vybranými vrcholy větší, než polovina celkové váhy, algoritmus končí a graf je rozdělen. Z obrázku 18 lze tedy určit, že spodní oblast grafu vybarvená modře obsahuje více než polovinu součtu váh hran celého grafu. Na obrázku 19 lze vidět přiblíženou oblast z předchozího grafu. K zlepšení tohoto rozdělení pomocí K-L algoritmu nedošlo z důvodu časové náročnosti tohoto algoritmu ($O(n^3)$). Tedy tento algoritmus nebyl aplikován na výsledné rozdělení grafu Edison.

Druhým testovaným byla mesh Metis_mesh. Tato velká síť byla rozdělena za pouhých 50 sekund. Bohužel nešlo na ni aplikovat K-L algoritmus pro zlepšení rozdělení této mesh. Z tabulky 8 lze také vyčíst, že algoritmus musel odstranit 5509 hran, aby rozdělil graf na dvě části. K-L algoritmus by v této situaci zřejmě nepomohl, jelikož ohodnocení hran v mesh je všude 1. Na obrázku 20 lze vidět výsledek tohoto rozdělení. Vzhledem k množství vrcholů slouží tento obrázek pouze jako ilustrační.

Posledním testovaným grafem byl graf Graf100. V tabulce 8 lze vidět, jak aplikováním K-L na původní rozdělení pomocí GGGP se minimální řez zmenšil, MaxMinCut index také a vzrostla modularita. Což nám říká, že kvalita dělení se zlepšila. Na obrázku 21 lze vidět rozdělení grafu těsně po aplikaci GGGP algoritmu. Na obrázku 22 lze vidět, co se stalo s rozdělením po aplikaci K-L algoritmu.



Obrázek 14: Kolize při dělení

Jelikož při dělení grafu pomocí GGGP může dojít k izolování části grafu patřícího do podgrafu G2 podgrafem G1 (viz obrázek 14), může dojít ke kolizi, kdy aplikováním K-L na takto nesouvislé podgrafy dojde ještě k většímu znehodnocení výsledku (viz obrázek 22). Možným řešením by bylo přidání takto izolovaných části podgrafu k druhému podgrafu, čímž by se odstranila nesouvislost podgrafů. Aplikovaný K-L algoritmus na takto upravené podgrafy by poté pracoval správně. Tato úprava byla provedena a výsledky lze vidět v tabulce v řádku s názvem Graf100_upd. Na obrázku 23 je taktéž výsledek těchto úprav zachycen. Podle tabulky 8 se po úpravě a aplikování K-L algoritmu minimální řez výrazně zmenšil, tedy došlo ke zlepšení kvality dělení grafu.

4. Testování GGGP algoritmu s vrcholem s největším stupněm.

U druhého testu dělení grafů pomocí algoritmu GGGP byl určen (jako počáteční vrchol) vrchol s největším stupněm. Dělení grafu Edison proběhlo velmi rychle - 1 min. Výsledek tohoto dělení lze vidět na obrázku 24. Výsledné rozdělení má velmi vysokou modularitu a nízký MaxMinCutIndex (viz tabulka 9). Z tohohle výsledku lze tedy usuzovat, že tento algoritmus se hodí na velmi velké grafy (oproti F-F algoritmu).

Název grafu	**Počet vrcholů**	**Počet hran**	**Čas dělení grafu [00:00:000]**	**Počet vrcholů podgrafu G1**	**Počet hran podgrafu G1**
Edison	2133	51285	01:00:873	1083	29397
Graf100	109	154	00:00:015	80	109
Graf100_upd	7434	22302	00:00:031	82	104
Metis_mesh	7434	22302	01:18:593	7251	21675

Počet vrcholů podgrafu G2	**Počet hran podgrafu G2**	**Minimální řez po GGGP**	**MaxMin-Cut po GGGP**	**Modularita po GGGP**	**Počet změn po K-L**	**Minimální řez po K-L**	**MaxMin-Cut po K-L**	**Modularita po K-L**
1050	14821	1319214	0,31017	0,49999	-	-	-	-
29	33	41	0,3737	0,49674	29	48	0,0242	0,4779
27	32	40	0,36021	0,49672	27	25	0,21368	0,49716
183	13	614	41,2343	0,499429	-	-	-	-

Tabulka 9: Výsledek testů GGGP s 2 vrcholy o nejvyšších stupních v grafu

Dalším děleným grafem byla mesh. Dělení proběhlo velmi rychle - 1:18 min. Výsledný MaxMinCut index je však velmi vysoký, což naznačuje, že mezi podgrafy existuje spousta vazeb. Také minimální řez to potvrzuje - 614. Tento algoritmus je pro tuto síť vhodnější, než F-F algoritmus, který oddělí zdroj od zbytku grafu (vzhledem k hustotě hran a ohodnocení).

Posledním testovaným grafem byl graf Graf100. U toho grafu, stejně jako v předchozím testování, došlo ke kolizi. V tabulce 9 pod názvem grafu Graf100_upd lze vidět výsledky upraveného grafu.

5. Dělení grafů pomocí aplikace Metis.

Název grafu	Počet vrcholů	Počet hran	Čas dělení grafu [00:00:000]	Počet vrcholů podgrafu G1	Počet hran podgrafu G1
Edison	2133	51285	00:00:016	1066	17647
Graf100	109	154	00:00:000	55	76
Metis_mesh	7434	22302	00:00:063	3717	9179

Počet vrcholů podgrafu G2	Počet hran podgrafu G2	Minimální řez po GGGP	MaxMin-Cut po GGGP	Modularita po GGGP	Počet změn po K-L
1067	29601	617745	1,11979	0,49999	-
54	73	10	0,0359	0,49829	0
3717	9189	3934	0,42704	0,49961	-

Minimální řez po K-L	MaxMin-Cut po K-L	Modularita po K-L
-	-	-
10	0,0359	0,49829
-	-	-

Tabulka 10: Výsledek testů z aplikace METIS

Posledním testovacím případem bylo provedení testů dělení grafů v programu METIS. Jelikož tento program nemá v sobě zahrnut F-F algoritmus, testování se omezilo pouze na GGGP algoritmus. Naměřené hodnoty tedy budeme porovnávat především s vlastní implementaci GGGP algoritmu. Také však bude zajímavé porovnat kvalitu rozdělení na daných grafech mezi F-F algoritmem a výsledkem z aplikace METIS.

Prvním testovaným grafem byl Graf100 (viz 25). Na obrázku 15 lze vidět výstup z aplikace METIS. Výpočet proběhl velmi rychle a na obrázku lze vidět, že ho program ani jakoby nezaregistroval. Výpočet proběhl pomocí GGGP algoritmu. Program METIS také nejdřív před výpočtem „zkrátí“ výsledný graf a pak provede výpočet pomocí GGGP a následně aplikuje výsledek na původní nezkrácený graf. Tedy výsledné časy budou z aplikace mnohem lepší. V porovnání s výsledky GGGP algoritmu, jak pro nejvzdálenější vrcholy, tak pro vrcholy s nejvyšším stupněm, vychází s nejlepší modularitou a MaxMinCut indexem výsledek z METIS aplikace. Také minimální řez je mnohem menší. V porovnání s F-F algoritmem však má vyšší minimální řez a horší MaxMinCut index. Modularita je téměř totožná. Tento fakt nasvědčuje tomu, že F-F algoritmus je pro tuto velikost grafů vhodnější, než GGGP algoritmus.

```
C:\tp\programs\Debug>gpmetis Graf100.GRAPH 2 -iptype=grow
**********************************************************************
METIS 5.0 Copyright 1998-11, Regents of the University of Minnesota
 (HEAD: , Built on: Mar 24 2013, 18:05:37)
 size of idx_t: zubits, real_t: zubits, idx_t *: zubits

Graph Information ---------------------------------------------------
 Name: Graf100.GRAPH, #Vertices: 109, #Edges: 154, #Parts: 2

Options -------------------------------------------------------------
 ptype=kway, objtype=cut, ctype=shem, rtype=greedy, iptype=metisrb
 dbglvl=0, ufactor=1.030, minconn=NO, contig=NO, nooutput=NO
 seed=-1, niter=10, ncuts=1

Direct k-way Partitioning -------------------------------------------
 - Edgecut: 5, communication volume: 10.

 - Balance:
     constraint #0:  1.028 out of 0.018

 - Most overweight partition:
     pid: 0, actual: 56, desired: 54, ratio: 1.03.

 - Subdomain connectivity: max: 1, min: 1, avg: 1.00

 - Each partition is contiguous.

Timing Information --------------------------------------------------
  I/O:                          0.000 sec
  Partitioning:                 0.000 sec   (METIS time)
  Reporting:                    0.000 sec

Memory Information --------------------------------------------------
  Max memory used:              0.038 MB
**********************************************************************
```

Obrázek 15: Výsledný výpis z aplikace METIS

Druhým testovaným grafem byl Edison. Tento graf rozdělila aplikace METIS na dvě stejné části s celkovým časem výrazně menším, než u testovací aplikace. Také minimální řez je téměř dvakrát menší. Výsledná modularita je mnohem větší, než u výsledků z testovací aplikace. Výsledky dělení tohoto grafu pomocí F-F algoritmu bohužel nejsou, tudíž nemůžeme porovnat.

Posledním testovaným byla mesh Metis_mesh. Aplikace METIS rozdělila tuto síť přesně na stejný počet vrcholů. F-F algoritmus tuto síť rozdělil velmi špatně, kdy oddělil zdroj od zbytku grafu. Výsledek dělení pomocí GGGP algoritmu testovací aplikací vyšel, co se týče kvality hůře, než tento výsledek z aplikace METIS. Především výsledný MaxMinCut index vyšel daleko menší než u testovací aplikace.

6.4 Vyhodnocení testů

V této podkapitole budou popsány a vyvozeny závěry z již naměřených a ukončených testů, které byly popsány v předešlé podkapitole a výsledky vizualizovány (viz příloha A). Výsledné shrnutí, pro jaký typ grafu je vhodný daný algoritmus, je přehledně popsán v tabulce 11.

Bylo prováděno celkem 5 různých testů na dvou algoritmech - GGGP a F-F algoritmus. Testy se prováděly vždy na třech stejných a různě velkých grafech. Prvním testovaným grafem byl Edison s přibližně 2000 vrcholy. Dále graf Graf100 s zhruba 100 vrcholy a Metis_mesh s okolo 7000 vrcholy. Metis_mesh byla použita z aplikace METIS. Bylo tedy třeba převést tento soubor typu MESH do testovací aplikace a z té následně do souboru

typu GDF. Také bylo třeba převést grafy Edison a Graf100 do souboru podporujícího aplikace METIS. Výsledek z programu METIS měl také jinou strukturu, kterou bylo třeba zpracovat a převést do GDF pro vizualizaci výsledků.

Z naměřených hodnot pro F-F algoritmus vyplývá, že je vhodný z časového hlediska spíše na menší grafy (obsahující stovky vrcholů). Také se nehodí pro dělení mesh. Vzhledem k hustotě hran této sítě se většinou stane, že se graf rozdělí na zdroj nebo spotřebič a druhou část tvoří zbytek grafu. Řešením tohoto problému by mohlo být omezení na počet vrcholů daného podgrafu. Výborně, jak po stránce časové, tak po stránce kvality pracoval tento algoritmus na grafu Graf100 (viz 6). Rozdělil tento graf na dva podgrafy s minimálním možným řezem (viz obrázek 16). Toho bylo ověřeno aplikováním K-L algoritmu na daný výsledek. Nedošlo již k žádnému přehození vrcholů. Stejného výsledku dosáhl jak s nastavením pro nejvzdálenější poč. vrcholy, tak i pro vrcholy s nejvyšším stupněm. Na grafu Edison se nepovedlo tento algoritmus otestovat z důvodu jeho časové složitosti.

Druhým algoritmem, který je stejně jako F-F algoritmus součástí testovací aplikace byl GGGP algoritmus. U tohoto algoritmu se také nastavoval počáteční parametr. Nastavil se jeden z vrcholů, jenž byl od toho druhého nejvzdálenější a také vrchol s nejvyšším stupněm. Při testování na třech různě velkých grafech se dospělo k výsledkům, že GGGP algoritmus nedosahuje takové kvality rozdělení u grafu Graf100 jako F-F algoritmus. Dokonce u GGGP algoritmu může dojít ke kolizi, kdy jeden z podgrafů izoluje část grafu (viz 14). Tento problém byl vyřešen algoritmem pro hledání dvou největších souvislých části, kdy kolizní části grafu byly přiřazeny jednomu z podgrafů (viz 23). Došlo tak ke korektnějším výsledkům a obzvlášť výsledkům po aplikování K-L. Tento algoritmus byl touto kolizí nejvíc ovlivňován a výsledky udával velmi nepřesné (viz 22). Dané výsledné rozdělení z hlediska kvality bylo velmi špatné. Po aplikaci K-L algoritmu na toto rozdělení došlo k mnoha úpravám a i přesto nedosáhlo toto rozdělení na kvalitu F-F algoritmu.

Tento algoritmus si však oproti F-F dokázal poradit s grafem Edison. V jednom případě to zvládl za více než 2 minuty a v druhém za 1 minutu. U výsledného zobrazení se pak ukázalo (viz 18), že spodní část grafu obsahuje více než polovinu váh hran grafu.

Posledním testovaným byla mesh s názvem Metis_mesh. GGGP algoritmus zvládl časově velmi rychle rozdělit tuto síť. Vzhledem však k hustotě hran tvoří jeden podgraf většina vrcholů. Pro dělení mesh by bylo vhodnější tento algoritmus upravit spíše na limit počtu vrcholů, než váh hran. Tak jak je tomu v aplikaci METIS.

Posledním krokem testování bylo tyto naměřené výsledky porovnat s výsledky obdržené z programu METIS. V této aplikaci byly testovány tři stejné grafy jako v předešlých testech. U každého výsledku byla doba testování velmi nízká a oproti testovací aplikaci zanedbatelná. Bylo tomu tak z důvodu, jak tato aplikace pracuje. Daný graf nejdříve „zkrátí“. Tento graf pak obsahuje velmi málo vrcholů. Na takto upravený graf aplikuje algoritmus (v našem případě GGGP). Výsledný řez aplikuje na původní graf. Tímto aplikace dosáhne velmi rychlého řešení. Druhým důvodem může být také lepší optimalizace aplikace, než je naimplementována vlastní testovací aplikace.

Prvním testovaným grafem byl Edison. Jelikož pomocí F-F algoritmu se nepovedlo tento graf rozdělit, můžeme srovnat výsledky jen s výsledky pomocí GGGP algoritmu.

Výsledky z aplikace METIS a výsledky z testovací aplikace jsou velmi podobné, ne-li totožné. Jelikož byla zvolena méně přesná metoda (dělení dle váh hran), než u aplikace METIS (dělení dle vrcholů), je konečné rozdělení testovací aplikací méně přesné. Kdybychom zvolili stejnou metodu, výsledek by byl totožný jen s časovým rozdílem. Proto byla zvolena na porovnání jiná varianta dělení.

Druhým testovaným grafem byl graf Graf100 (viz 25). Jestliže porovnáme výsledek z aplikace METIS a GGGP algoritmem v testovací aplikaci, zjistíme, že výsledky jsou si taktéž velmi podobné. Kvalitou je nepatrně lepší výsledek z aplikace METIS, ale minimální řez je větší, než u výsledku z testovací aplikace při použití GGGP algoritmu s nevzdálenějším vrcholem. U druhého testu GGGP s největším stupněm vrcholu je výsledek výrazně horší. Porovnáním kvality výsledku s F-F algoritmem zjistíme, že výsledné rozdělení pomocí F-F je kvalitnější, má menší minimální řez, než z aplikace METIS. Tedy pro tyto středně velké grafy je vhodnější F-F algoritmus, než GGGP.

Posledním porovnávaným je síť Metis_mesh. F-F algoritmus je naprosto nevhodný pro tuto síť, tedy výsledky budeme porovnávat jenom s GGGP v testovací aplikaci. Při porovnání těchto výsledků, lze pěkně vidět rozdíl obou metod. Jelikož síť je velmi hustě propojena, tak výsledek z testovací aplikace vrátí jeden podgraf s většinou vrcholů a druhý podgraf s minimem vrcholů. U GGGP algoritmu se zvoleným nejvyšším stupněm vrcholů je výsledek ještě patrnější. Kdežto algoritmus z aplikace METIS rozdělí síť přesně na polovinu. Porovnávat tedy tyto výsledky nelze.

V tabulce 11 nyní vidíme graficky znázorněné, které algoritmy, na kterých grafech obstály a naopak, které neobstály.

Název grafu	**Velký graf (Edison)**	**Malý graf (Graf100)**	**Mesh (Metis_mesh)**
F-F alg. s 2 nejvzdálenějšími vrcholy	✗	✓	✗
F-F alg. s 2 vrcholy nejvyššího stupně	✗	✓	✗
GGGP alg. s nejvzdálenějším vrcholem	✓	✗	✓
GGGP alg. s vrcholem nejvyššího stupně	✓	✗	✗
GGGP alg. z aplikace METIS	✓	✓	✓

Tabulka 11: Přehled vhodnosti testovaných algoritmů na typy grafů

7 Závěr

Cílem této práce bylo získat základní přehled o existujících řešeních pro dělení grafů, popsat tyto algoritmy řešící tento problém a v praktických experimentech ověřit jejich kvalitu a časovou náročnost. Během této práce jsem se seznámil se základními principy a vlastnostmi jednotlivých algoritmů. Také jsem se seznámil s využitím těchto algoritmů v praxi - doprava, sociální sítě apod. Tyto informace a mnohé jiné jsem uvedl v této práci.

Druhým cílem mé práce bylo vytvořit testovací aplikaci, která bude obsahovat alespoň dva algoritmy na dělení grafů. Tato aplikace byla vytvořena za účelem porovnání výsledků oproti aplikaci METIS s kterou jsem byl seznámen. Úkolem bylo také zjistit, zda kvalita a čas výsledného dělení pomocí Ford-Fulkersnova algoritmu je lepší, než výsledek z aplikace METIS.

Poté, když byla testovací aplikace hotova, byly navrženy testy za účelem měření kvality a časové náročnosti jednotlivých algoritmů. Každý algoritmus byl testován na třech stejných grafech, aby bylo možné posléze porovnat naměřené výsledky. Celkem proběhlo 15 testů na třech grafech o různé velikosti. Výsledky těchto testů byly zaznamenány do tabulek a také vizualizovány programem Gephi.

Nakonec proběhlo vyhodnocení a porovnání jednotlivých výsledků mezi sebou a mezi aplikaci METIS. Z těchto výsledků byly posléze vyvozeny závěry dle cílů této práce.

Další rozvoj této aplikace bych viděl v implementaci hierarchického dělení grafů na více, než jen dvě části. Také by se aplikace mohla rozšířit o další algoritmy, které obsahuje aplikace METIS a následně porovnávat dosažené výsledky.

8 Reference

[1] Brázdová, Markéta. *Využití Některých metod teorie grafů při řešení dopravních problémů*. 2007. [online]. [cit. 2013-01-06]. Dostupné z: http://pernerscontacts.upce.cz/05_2007/Brazdova.pdf.

[2] Houser, Pavel. *Svět na pouhých šest kroků*. 2003. [online]. [cit. 2013-01-06]. Dostupné z: http://scienceworld.cz/technologie/svet-na-pouhych-sest-kroku-2780.

[3] FORTUNANTO, Santo. *Community detection in graphs*. Complex Networks and Systems Lagrange Laboratory,Torino. Dostupné z: http://arxiv.org/pdf/0906.0612.pdf.

[4] DEMEL, Jiří. *Grafy a jejich aplikace*. Vyd. 1. Praha: Academia, 2002, str. 129 - 145. ISBN 80-200-0990-6.

[5] KOVÁCS, Ferenc, LEGÁNY, Csaba, BABOS, Attila. *Cluster Validity Measurement Techniques*. Department of Automation and Applied Informatics Budapest University of Technology and Economics Goldmann György, Budapešť. [online]. [cit. 2013-01-06]. Dostupné z: http://www.academia.edu/735094/Cluster_validity_measurement_techniques.

[6] Halkidi, Maria, Batistakis, Yannis, Vazirgiannis, Michalis. *Clustering Validity Checking Methods: Part II*. Department of Informatics, Athens University of Economics & Business. [online]. [cit. 2013-01-06]. Dostupné z: http://libra.msra.cn/Publication/56877/clustering-validity-checking-methods-part-ii.

[7] DELLING, Daniel, GAERTLER, Marco, GÖRKE, Robert, NIKOLOSKI, Zoran, WAGNER, Dorothea. *How to Evaluate Clustering Techniques* Faculty of Informatics, Universität Karlsruhe. [online]. [cit. 2013-01-06].

[8] *Klasifikace problémů*. [online]. 2011. [cit. 2013-02-02]. Dostupné z: http://www.512.cz/index.php?title=Klasifikace_probl%C3%A9m%C5%AF.

[9] Kačmarik, Krivák, Remiš, Kozák, Tůma. *NP-úplné problémy*. [online]. 2010. [cit. 2013-02-02]. Dostupné z: http://mj.ucw.cz/vyuka/0910/ads2/11-np.pdf.

[10] Valla, Tomáš, Matoušek, Jiří. *Kombinatorika a grafy I.* Matematicko-fyzikální fakulta, Karlova univerzita. 28.5. 2008. [online]. [cit. 2013-02-05]. Dostupné z: http://kam.mff.cuni.cz/ṽalla/kg.pdf.

[11] *Finite Elements Method Software Development*. © 2009 [online]. [cit. 2013-02-05]. Dostupné z: http://www.natsakis.com/?q=node/11.

[12] Zienkiewicz, O. C. *The Finite Element Method: Its Basis and Fundamentals*, Sixth Edition. Butterworth-Heinemann, 2005. [online]. [cit. 2013-02-07].

[13] *Gephi* © 2008-2012 [online]. [cit. 2013-02-15]. Dostupné z: https://gephi.org.

[14] *METIS* © 2006-2011 [online]. [cit. 2013-02-20]. Dostupné z: http://glaros.dtc.umn.edu/gkhome/views/metis.

[15] László, Lovász, Korte, Prömel, Schrijver. *Network Flow Algorithms.* Paths, Flows and VLSI-Layout. Springer-Verlag Berlin Heidelberg. 1990. [online]. [cit. 2013-02-25]. Dostupné z: http://www.cs.cornell.edu/people/eva/Network.Flow.Algorithms.pdf.

[16] Kovář, Petr. *Úvod do Teorie grafů*. Vysoká škola báňská – Technická univerzita Ostrava a Západočeská univerzita v Plzni. 2012. [online]. [cit. 2013-02-12]. Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/uvod_do_teorie_grafu.pdf.

[17] Kovář, Petr. *Teorie grafů*. Vysoká škola báňská – Technická univerzita Ostrava a Západočeská univerzita v Plzni. 2012. [online]. [cit. 2013-02-12].

[18] LOVÁSZ, László. *Random Walks on Graphs: A Survey.* Combinatorics. 1993. [online]. Keszthely (Maďarsko) [cit. 2013-03-12]. Dostupné z: http://www.cs.elte.hu/ lovasz/erdos.pdf.

[19] LOVÁSZ, László. *Nezávislé množiny*. © 2011 [online]. [cit. 2013-02-15]. Dostupné z: http://www.algoritmy.net/article/7303/Nezavisle-mnoziny–Vrcholove-pokryti

[20] Youssef, Habib, Sait, Sadiq. *Partitioning*. King Fahd University of Petroleum & Minerals. 2003 [online]. [cit. 2013-03-25]. Dostupné z: http://www.scribd.com/doc/7203614/11/Kernighan-Lin-Algorithm

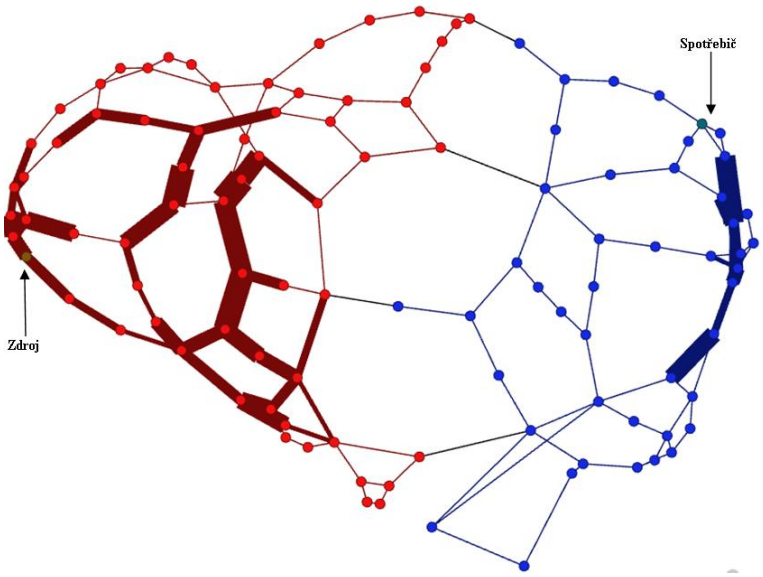
[21] Karypis, George, Kumar, Vipin. *METIS**. University of Minnesota. 1995 [online]. [cit. 2013-01-22].

[22] Spielman, Daniel, Teng, Hua Shang. *Spectral partitioning works: Planar graphs and finite element meshes*. University of Minnesota. [online]. [cit. 2013-02-03].

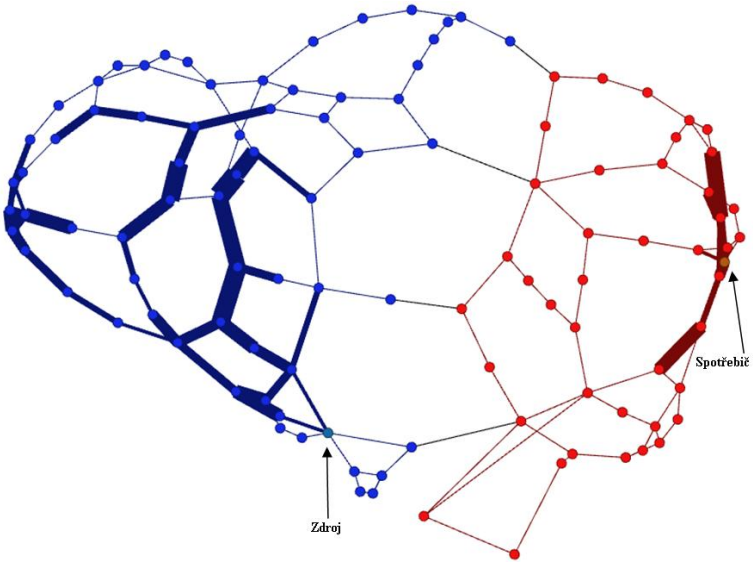
[23] Hendricson, Bruce, Leland, Robert. *A multilevel algorithm for partitioning graphs*. Sandia national laboratories. [online]. [cit. 2013-03-13].

A Grafy

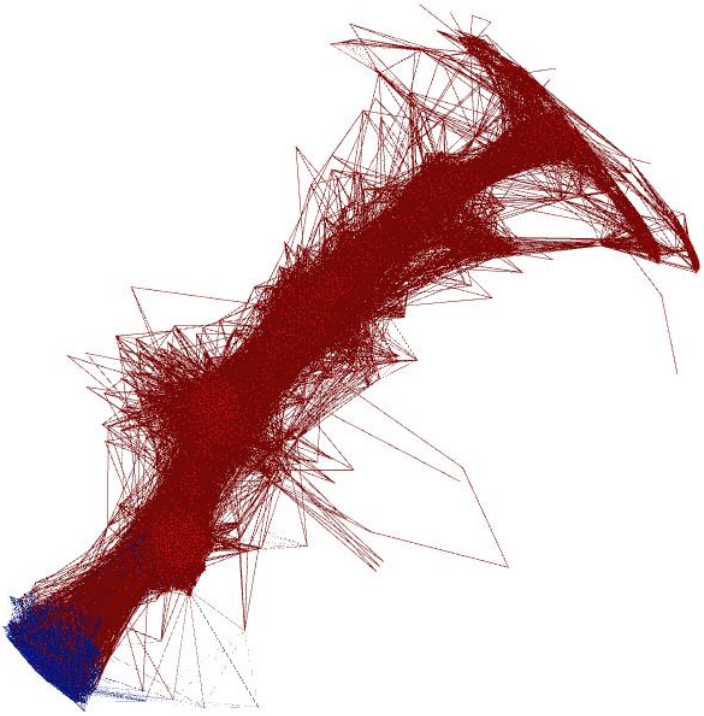
Zde jsou umístěny grafy, které zachycují výsledky jednotlivých testů z vlastní a METIS aplikace. Jedná se o grafy týkající se testů popsaných v kapitole 6.



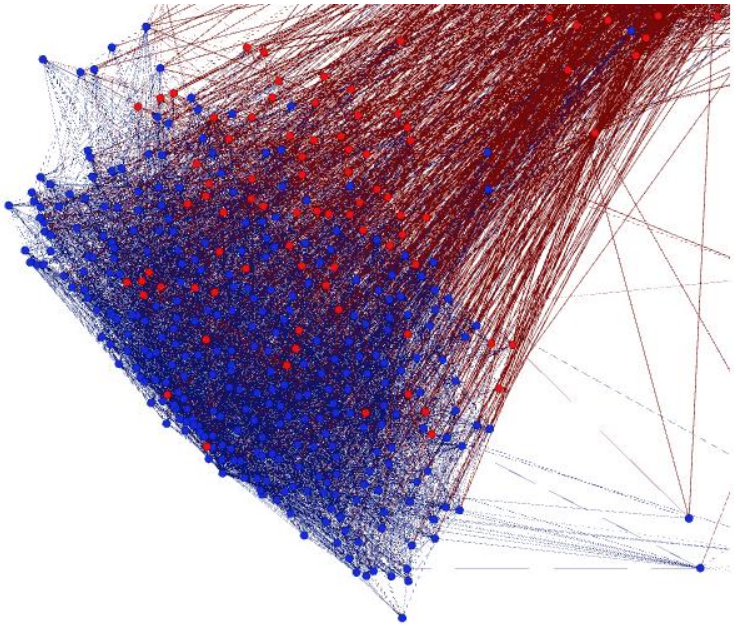
Obrázek 16: Výsledné rozdělení grafu Graf100 pomocí F-F algoritmu



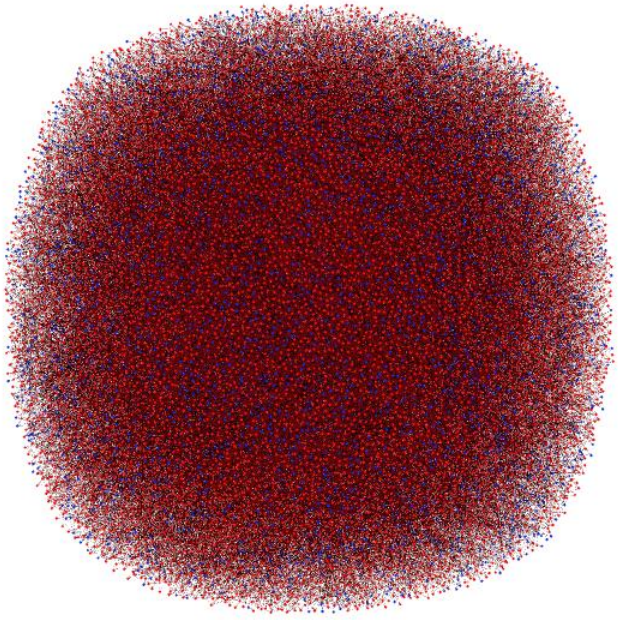
Obrázek 17: Výsledek po rozdělení grafu Graf100 pomocí F-F algoritmu



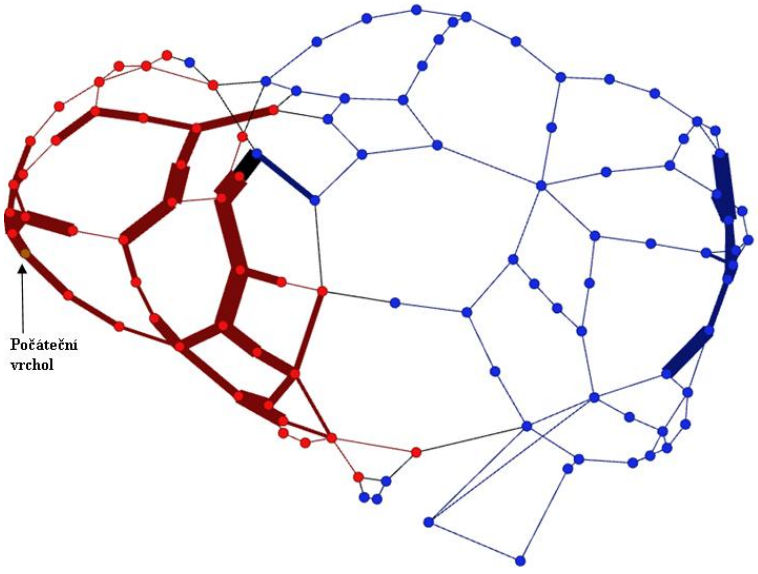
Obrázek 18: Výsledek po rozdělení grafu Edison pomocí GGGP algoritmu



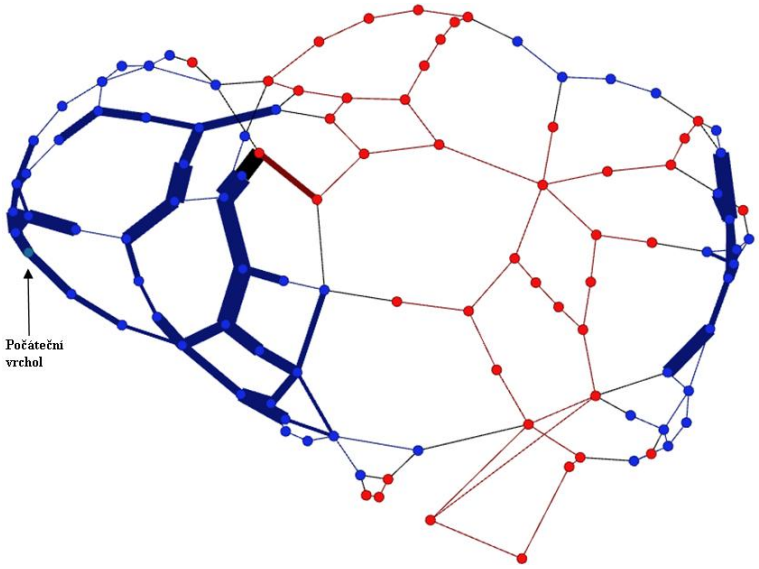
Obrázek 19: Přiblížení výsledného grafu Edison rozděleného pomocí GGGP algoritmu



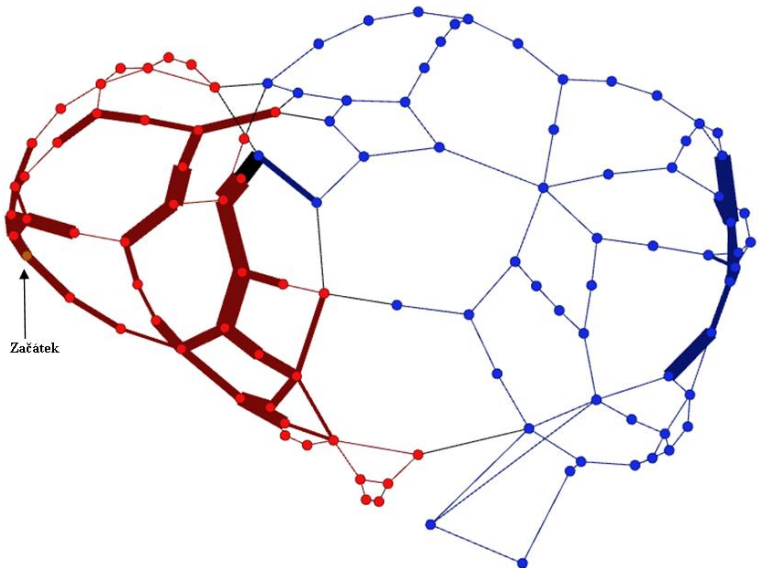
Obrázek 20: Výsledek rozdělení mesh pomocí GGGP algoritmu



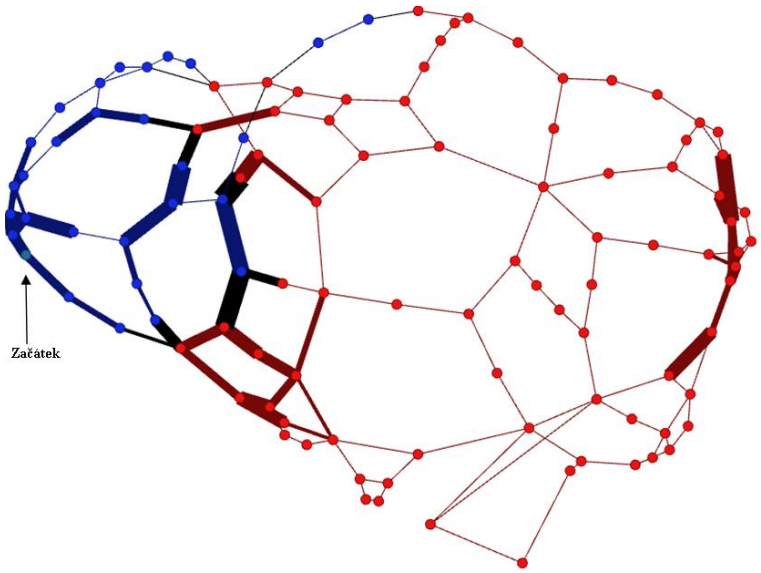
Obrázek 21: Výsledek rozdělení Graf100 pomocí GGGP algoritmu



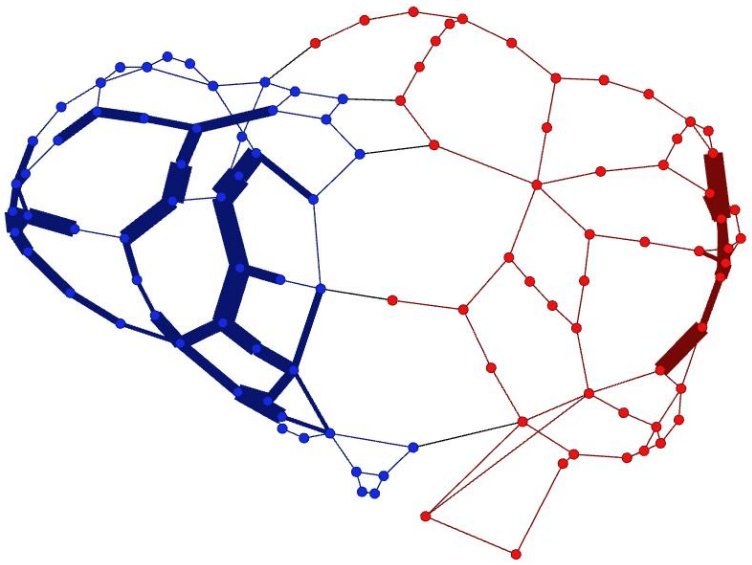
Obrázek 22: Výsledek rozdělení Graf100 pomocí K-L algoritmu



Obrázek 23: Výsledek rozdělení Graf100 po úpravě



Obrázek 24: Výsledek rozdělení grafu Graf100 pomocí GGGP



Obrázek 25: Výsledek rozdělení grafu Graf100 pomocí GGGP

B Obsah přiloženého CD

- Diplomová práce v elektronické podobě
- Testovací aplikace pro vybrané dělící algoritmy
- Upravená verze aplikace METIS
- Testovací grafy používané v testech